2015

Multi-directional Warning Message Dissemination Protocol Based on Motion Vector Clustering for VANETs

Hung-Chou Wang
Andrews University, wang@andrews.edu

ABSTRACT

MULTI-DIRECTIONAL WARNING MESSAGE DISSEMINATION PROTOCOL BASED ON MOTION VECTOR CLUSTERING VANETS

by

Hung-Chou Wang

Chair: Roy Villafane, Ph.D.

ABSTRACT OF GRADUATE STUDENT RESEARCH

Thesis

Andrews University

College of Arts and Sciences

Title: MULTI-DIRECTIONAL WARNING MESSAGE DISSEMINATION PROTOCOL BASED ON MOTION VECTOR CLUSTERING FOR VANETS

Name of researcher: Hung-Chou Wang

Name and degree of faculty chair: Roy Villafane, Ph.D.

Date completed: July 2015

Problem

Most broadcast suppression protocols in vehicular ad hoc networks (VANET) mainly focus on one-dimensional message dissemination model for both highway and urban scenarios. Due to the non-line-of-sight (NLOS) problem occurring frequently in urban scenario, protocols mostly rely on either infrastructure or the vehicle that is passing through the intersection to forward the message in multiple directions manner. However, these one-dimensional message dissemination models fail to take into account realistic road topologies and traffic distribution. As a result, they tend to miss some possible dissemination directions.

Method

Vehicles travelling on the same road share similar motion pattern due to the constraint of road topology. Each motion pattern represents a road topology as well as a potential dissemination direction. By identifying motion pattern of one-hop neighbors, the proposed motion vector protocol (MVP) enables a vehicle not only to identify potential dissemination directions without the support from infrastructure or a road map but also to make suppression decisions without any additional information from periodic beacons.

Results

The total number of transmissions for simple flooding (each node broadcasts once) compared with MVP ranges respectively as follows: 90.2-269.7 and 40.6-72.3. Also, the number of saved rebroadcasts for simple flooding compared with MVP ranges respectively as follows: 0%-0% and 57%-73%. In the case of reachability, the simple flooding compared with MVP ranges 100%-100% and 100%-100% respectively. Finally, the average latency of the entire dissemination for simple flooding and MVP ranges 0.01446-0.01286s and 0.1127-0.1565s respectively.

Conclusions

The experimental results show that MVP achieves high reachability, while still significantly reducing rebroadcast redundancy. One distinctive feature of MVP is that it is capable of operating on complex road topology such as a

roundabout, curve road, branch road, etc., with multi-directional traffic in it.

Andrews University

College of Arts and Sciences

MULTI-DIRECTIONAL WARNING MESSAGE DISSEMINATION PROTOCOL BASED ON MOTION VECTOR CLUSTERING FOR VANETS

A thesis

Presented in Partial Fulfillment

of the Requirement for the Degree

Master of Science

by

Hung-Chou Wang

2015

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS iv

LIST OF ABBREVIATIONS vi

Chapter
1. INTRODUCTION 1

2. RELATED WORK 4

The Last One Protocol (TLO) 4
Adaptive Probability Alert Protocol (APAL) 4
Acknowledged Broadcast from Static to Highly Mobile Protocol (ABSM) 5
Urban Multi-hop Broadcast Protocol (UMB) 6
Broadcast Suppression Techniques (BST) 7
Distributed Optimized Time Slot Protocol (DOT) 9
Adaptive Multi-Directional Data Dissemination Protocol (AMD) 10

3. THE PROTOCOL 15

Requirements and Assumptions 18
Message Structure 18
Data Collection & Preparation 19
Motion Vector Clustering Scheme 19
a. Computing Distance Matrix 21
b. Removing Redundant Paths 24
c. Clustering Vectors 25
Boundary Similarity Verification (BSV) ... 28
d. Processing Clustering Results 30
Rebroadcast List Construction 31
a. Sorting Each Cluster by its Own Resultant Vector 32
b. Selecting Two Outermost Vehicles from Each Cluster to Construct Rebroadcast List 34
c. Appending Isolated Vector List Into Rebroadcast List 35
d. Sorting Rebroadcast List with Vector or Distance Metrics 37
Delay-Based Suppression Scheme 41
Intra-Cluster and Global Cancellation 44

4. SIMULATION RESULTS 51

5. CONCLUSION 61

BIBLIOGRAPHY 62

LIST OF ILLUSTRATIONS

1. Uneven Traffic Distribution Problems in BST 8

2. Multi-Directional Broadcasting Scheme in AMD 13

3. The Flow Chart of the MVP 17

4. The Illustrations of Proposed Graph Model 21

5. The Illustrations of Spatial Distance between Two Flow Vectors in Different Formation..... 23

6. Clustering Problem When Only Relying on Forward Distance..... 26

7. The Clustering Algorithm of MVC 27

8. The Illustration of BSV 29

9. The Illustration of the Rebroadcast List Construction..... 32

10. The GPS Coordinate System (Datum) Used by MVP 33

11. The Illustrations of Propagation Problems in Distance-Based Schemes..... 36

12. The Illustration of Isolated Vector 36

13. The Illustration of Sorting Algorithms in MVP 38

14. The Delay-Based Suppression Scheme Used by MVP 42

15. The Algorithm of the Intra-Cluster Cancellation 45

16. The Map Fragment Used in Simulation 51

17. Simulation Result of Total Transmission Number (Vector Sorting)..... 53

18. Simulation Result of Save Rebroadcast (Vector Sorting)..... 54

19. Simulation Result of Reachability (Vector Sorting)..... 55

20. Simulation Result of Average Latency (Vector Sorting)..... 56

21. The Comparison of Total Transmission Number by Varying CF(i) (Vector Sorting) 57

22. The Comparison of Total Transmission Number by Varying CF(i) (Vector Sorting) 58

23. The Comparison of Total Transmission Number by Varying CF(i) (Distance Sorting) 58

24. The Comparison of Average Latency by Varying CF(i) (Distance Sorting) 59

25. Reachability Issue When Adopting Distance Sorting ... 60

LIST OF ABBREVIATIONS

ABSM	Acknowledged Broadcast from Static to Highly Mobile Protocol
ACK	Acknowledgement
AMD	Adaptive Multi-Directional Data Dissemination Protocol
APAL	Adaptive Probability Alert Protocol
BHL	Back Half of the Neighbor List
BSM	Basic Safety Message
BST	Broadcast Suppression Techniques
BT	Received Beacons Table
BTC	A Copy of Received Beacons Table
CAM	Cooperative Awareness Message
CCH	Control Channel
CDS	Connected Dominating Set
CF	Candidate Forwarder
CF(i)	Candidate Forwarder Number
CTB	Clear to Broadcast
CTS	Clear to Send
DFV	Intermediate Database for MVC (Using Flow Vector ID to distinguish vehicles)
DID	Database for Message Dissemination & Cancellation modules (Using vehicle ID to distinguish vehicles)
DM	Distance Matrix
DMC	A Copy of Distance Matrix
DOT	Distributed Optimized Time Slot Protocol

DTL	Dissemination Targets List
FD	Forward Distance
FV	Flow Vector
FHL	Front Half of the Neighbor List
GPS	Global Positioning System
IVC	Inter-Vehicle Communication
ML	Members List
MPS	Motion Patterns Study
MT	Received Data Messages Table
MVC	Motion Vector Clustering Scheme
MVP	Motion Vector Protocol
MYL	Neighbors on My Side List
NES	Neighbor Elimination Scheme
NL	Neighbor List
NLOS	Non-Line-Of-Sight
OBU	On-Board Unit
OCA	Overlapping Coverage Area
OPL	Neighbors on Opposite Side List
ReC	Receiver Consensus Protocol
RSS	Received Signal Strength
RSU	Road Side Unit
RTB	Request to Broadcast
RTS	Request to Send
SPM	Shortest Path Matrix
SRL	Sender’s Rebroadcast List
SCL	Safe for Cancellation List
TLO	The Last One Protocol
UMB	Urban Multi-Hop Broadcast

VANET	Vehicular Ad hoc Network
WSMs	WAVE Short Messages

CHAPTER 1

Introduction

In the first part of the 21st century, GPS technology and wireless communication devices became more and more accessible to the general public. In the foreseeable future, vehicles equipped with inter-vehicle communication (IVC) devices will revolutionize many aspects of people's driving experiences, such as safety, transport efficiency, and infotainment. All of these are achievable by means of Vehicular Ad hoc Network (VANET). The primary goal for VANET is to provide safety related information in a timely manner to all reachable vehicles within a critical region, where multi-hop broadcasting is commonly used. However, as this relies on broadcasting as the communication medium, the broadcasting protocol has to be properly crafted or suitable mechanisms have to be introduced, otherwise, the network is prone to suffer from the broadcast storm problem (Tseng, Ni, Chen, & Sheu, 1999).

Therefore, many broadcast storm mitigation techniques have been proposed in the literature (Ros, Ruiz, & Stojmenovic, 2012; Suriyapaibonwattana & Pornavalai, 2008, 2009; Wisitpongphan, Tonguz, Parikh, Mudalige, Bai, & Sadekar, V, 2007). Although these approaches seem to be different, the main principle governing them is similar. This is done by selecting a minimum number of candidate forwarders (CFs) that

can cover the intended dissemination area. In one study (Wisitpongphan, et al., 2007), three of the most widely used schemes are proposed, i.e. weighted p-persistence, slotted 1-persistence,and slotted p-persistence, where the authors claim that one-dimensional model can well capture the topology of the network in the VANET context. In another study (Schwartz, Scholten, & Havinga, 2013), an improved scheme based on former schemes is proposed, where the authors address the need of multi-directional dissemination and proposed Adaptive Multi-directional data Dissemination (AMD) to cope with the dissemination problem in urban scenarios. However, none of them are capable of operating on complex road topology such as a roundabout, curve road, branch road, etc., with multi-directional traffic in it.

To this end, a Motion Vector Protocol (MVP) is proposed, which is designed for multi-directional scenario in VANET. The scheme tackles current insufficiencies by adopting the motion patterns study (Hu, Ali, & Shah, 2008) which enables vehicles to identify traffic flow in the vicinity. Based on this information, the proposed protocol is able to make better decisions on message dissemination and broadcast suppression without any assistance from a road map. Also, aside from a GPS receiver, electronic compass, and IEEE 802.11p commutation devices (they should be integrated into OBU, on-board unit), the protocol requires no extra hardware. Simulations show that MVP can achieve the same coverage as simple flooding (each node broadcasts once), while still significantly reducing redundant messages.

The remainder of this thesis is organized as follows. Chapter 2 introduces the related work in the literature regarding the broadcast suppression technique in VANET. The proposed scheme is presented in Chapter 3. Chapter 4 describes the simulation setup and then evaluates the performance. Lastly, Chapter 5 overs conclusions to the reader.

CHAPTER 2

Related Work

Extensive research and broadcast suppression protocols have been proposed in the literatures. This section includes techniques and protocols that are relevant to this work.

The Last One (TLO)

TLO aims to select only one vehicle which is the farthest away from the sender to forward the message (Suriyapaibonwattana & Pornavalai, 2008). The vehicles adapt this scheme by waiting a short interval after receiving a message to elect the most distant vehicle from the sender by sharing and comparing distance information (distance between itself to the sender) with neighboring vehicles. The elected vehicle will make three broadcast attempts before assuming that there is no vehicle behind it. As for the remaining vehicles, they will wait on a timer until their threshold arrives and then return back to their normal state. Clearly, this scheme is designed for a one-dimensional message dissemination scenario, and only suitable for highways.

Adaptive Probability Alert Protocol (APAL)

APAL makes the rebroadcast decision based on how many duplicate messages have been received during a random selected interval (Suriyapaiboonwattana, Pornavalai, & Chakraborty,

2009). Upon the interval expiration, the algorithm checks the number of duplicate messages and uses it to calculate rebroadcast probability (Pi+1) and wait time (Δτi+1). APAL categorizes four possible scenarios: 1.) not received duplicate, Pi+1 = 0.7-0.9 (high probability); 2.) received duplicates in Δτi, Pi+1 = Pi/duplicate number ; Δτi+1 = Δτi/duplicate number; 3.) not received duplicates, successful to rebroadcast, Pi+1 = Pi /2; Δτi+1 = Δτi; and 4.) not received duplicates, fail to rebroadcast, Pi+1 = Pi *2; Δτi+1 = Δτi /2.

There are three more variables which define the terminating conditions of the algorithm for exiting the process. The simulation result shows the robustness of the scheme since the performance will not degrade while increasing the number of vehicles.

Acknowledged Broadcast from Static to Highly Mobile Protocol (ABSM)

ABSM, a backbone approach, adopted the connected dominating set (CDS) scheme to find the minimum number of forwarders and the neighbor elimination scheme (NES) to eliminate the neighbor that confirms the message has been properly received (Ros, Ruiz, & Stojmenovic, 2012; Stojmenovic, 2004; Stojmenovic, Seddigh, & Zunic, 2002). The confirmation is done by attaching the acknowledgement (ACK) to a periodic beacon, where a vehicle obtains its neighbors' information to compute its CDS status and to maintain two neighbor lists (Received list R, and Not received list N). In ABSM, upon receiving a message for the first time, each

vehicle sets up a timer based on its CDS status (the vehicle belonging to CDS has a shorter wait time than those who do not belong to CDS). Once the timer expires, the vehicle checks whether list N is empty or not. If N is not empty, the vehicle retransmits the message.

Normally, these types of schemes are not suitable for delivering safety related messages due to high latency. Other variants such as Receiver Consensus (ReC) aim to elevate this drawback by introducing a ranking mechanism to determine rebroadcast priority locally (Liu, Yang, & Stojmenovic, 2013). The simulation result shows sufficient improvement in ReC.

Urban Multi-hop Broadcast (UMB)

Urban Multi-hop Broadcast (UMB), a MAC-based protocol, tackles broadcast storm and hidden terminal problems by introducing Request to Send/Clear to Send (RTS/CTS) handshake and acknowledgement (ACK) mechanism to the proposed scheme (Korkmaz, Ekici, Özgüner, & Özgüner, 2004). RTS/CTS handshake and ACK mechanism are not commonly used in broadcast protocol since multiple receivers may cause local broadcast storms around the sender or flood the network with traffic. Therefore, UMB utilizes the black-burst mechanism (channel jamming signal) to identify the farthest vehicle from the sender. Before the sender accurately broadcasts the message, it first broadcasts a Request to Broadcast (RTB) packet. Then all the vehicles in the dissemination direction reply to it with black-burst signals whose durations are proportional to their distance to the sender. Next, each receiver listens to the channel to verify whether it can detect any black-burst

signal still in the air. The receiver who detects no black-burst signal wins the chance to be the next hop forwarder by broadcasting a Clear to Broadcast (CTB) packet. After that, the sender attaches the ID of the vehicle who sent CTB into the message as ACK. Finally, the sender broadcasts the message. Additionally, in order to disseminate messages in an urban setting, the scheme assumes that each intersection is equipped with a repeater which helps to distribute messages to all possible directions, namely, directional broadcast. Other variants can be found in the literature (Korkmaz, Ekici, & Özgüner, 2006), where the repeater is replaced by the vehicle that is passing through the intersection.

Broadcast Suppression Techniques (BST)

BST aims to suppress the broadcast storm problem by means of three probabilistic and timer-based schemes along with their received signal strength (RSS) versions (Wisitpongphan, et al., 2007). These schemes include Weighted p-Persistence which is a probabilistic scheme where the forwarding probability of a receiver is based on the distance from its location to the sender. The farther they are, the higher probability is. The probability is defined as follows: $P_{ij} = D_{ij}/R$, where D is the distance between the receiver and the sender, and R is the sender's transmission range.

Another scheme is the Slotted 1-Persistence which is a timer-based scheme where a receiver calculates its wait timer based on the time slot equations given below, and rebroadcasts the message with probability 1 at assigned time slot if it

does not receive any duplicates. Time slot scheme segments senders' transmission range into several sub-segments, which allows vehicles within farther sub-segments to have a shorter wait time than those within closer sub-segments. Once the message has been successfully rebroadcasted, vehicles within this segment suspend their scheduled rebroadcast (since the rebroadcast pocket is a duplicate of the original message). The main concern of this scheme is that the number of slots (segment numbers) is a predetermined number which may not reflect traffic conditions properly (known as uneven traffic distribution problem) as illustrated in Figure 1.

$$Ts_{ij} = S_{ij}/\tau$$
$$S_{ij} = Ns\left\{1 - \left\lceil \frac{min(D_{ij}, R)}{R} \right\rceil\right\}$$

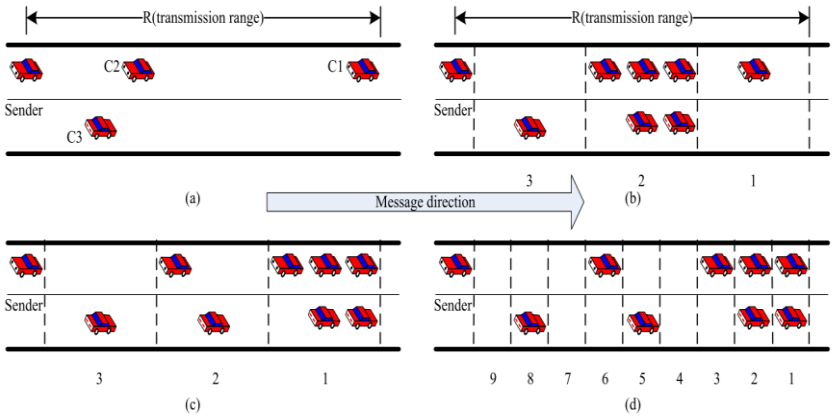


Figure 1. The illustration of Uneven Traffic Distribution Problems in BST (Li, Wang, Yao, & Chen, 2013)

Lastly, is the Slotted p-Persistence which is similar to Slotted 1-Persistence except a receiver broadcasts the message with a predetermined probability other than 1 at an assigned time slot. Also, the scheme demands all receivers buffer the message for a period of time. If the message has not been

heard by a receiver after the time passed, it rebroadcasts the message with a probability of 1 to prevent the message dying out. The equation for this is $S_{ij} = [Ns - 1] \times WAIT\_TIME + \delta ms$

BST only considers one-dimensional topology and uses distance as the metric. Therefore, it is not capable of multi-directional data dissemination.

Distributed Optimized Time Slot (DOT)

Distributed Optimized Time slot (DOT), a Slotted 1-Persistence based scheme, resolves the predetermined number problem in BST (Schwartz, Das, Scholten, & Havinga, 2012). Vehicles that share a similar distance to the sender will be assigned to the same time slot; they may attempt to rebroadcast simultaneously causing undesired message collisions and contentions. To cope with this problem, DOT suggests that first, each receiver obtains its one-hop neighbors' coordinates data through beacons and makes a list $\vec{v}$, where the neighbors outside the sender's transmission range are excluded. Then it sorts the list $\vec{v}$, by neighbors' distance to the sender. Next, the receiver finds its position on the list $\vec{v}$ and applies its position to the equation given below. After that, it sets up a timer based on the wait time obtained from the equation. Finally, it rebroadcasts the message upon the timer's expiration if it does not receive any duplicate messages. In this way, DOT is able to control the density of time slots precisely (reducing the number of vehicles sharing the same time slot). However, due to the position difference, receivers may detect different sets of one-hop neighbors in

their own transmission ranges (scopes). Therefore, they might not have a constant agreement on which vehicle is the farthest vehicle from the sender. Given the fact that DOT is a receiver-oriented approach (receivers decide whether to rebroadcast or not), the hidden terminal problem in DOT is inevitable. Equation one is known as $Ts_{ij} = st\left\{\left\lceil\frac{S_{ij}+1}{tsd}\right\rceil - 1\right\} + AD_{ij}$..

Adaptive Multi-Directional Data Dissemination Protocol (AMD)

A Slotted 1-Persistence based scheme is designed for both highway and urban scenarios (Schwartz, Scholten, & Havinga, 2013). In order to provide a full scale solution, the authors combined the Distributed Optimized Time slot (DOT) with the Simple and Robust Dissemination (SRD) scheme giving Adaptive Multi-Directional Data Dissemination (AMD) three unique features: adaptive multi-directional dissemination, time slot density control, and store-carry-forward.

Adaptive multi-directional dissemination is achieved by utilizing GPS data, attaching directional vector data to warning messages, and customizing road maps. The road map has to be pre-loaded to each vehicle and each center point of every intersection must be explicitly marked. Since AMD assumes one-dimensional topology for highway and Manhattan-Grid topology for urban scenario, it reduces the possible dissemination directions to two and four. By marking the center point of each intersection on the map, it allows the vehicle to identify the number of dissemination directions

when it is approaching the location (approximately 15). Also, AMD adapted the sender-oriented approach, where a sender designed a group of CFs to forward warning messages. This enables the sender to coordinate the rebroadcast sequence among CFs. This design helps not only to accelerate the dissemination but also to prevent the hidden terminal problem as stated above.

Time slot density control originates from DOT, which was designed to reduce the number of vehicles which are assigned to the same time slot to prevent simultaneous rebroadcasts. However, in AMD, it is used to assign vehicles in different directions to the same time slot, therefore, accelerating the dissemination.

Store-carry-forward initiates from SRD, which is designed to forward messages in a disconnected network, occurring when traffic density is low. The most distant vehicle in one of dissemination directions, is assigned the duty of buffering the message. Once it encounters other vehicles outside the sender's transmission range, it rebroadcasts the message.

In AMD, before a sender broadcasts a warning message, it obtains its one-hop neighbors' coordinates data through beacons and makes a list $\vec{v}$. Then it sorts the list $\vec{v}$ by the neighbors' distance to itself (farther neighbors are put on top of the list, and the arrangement follows a spiral shape pattern as shown in Figure 2(a)). Next, the sender attaches list $\vec{v}$ along with the directional vector data $\vec{a}$, b ($\vec{a}$ is the sender's velocity vector rotated by β degree, where $\beta = 360/2b$;

b is the number of direction sectors, as shown in Figure 2(c)) to the message. After that, it broadcasts the message.

When considering a receiver, it finds its position on the list $\vec{v}$, and applies its position to the equation 1. Then it sets up a timer based on the wait time obtained from the equation. Next, it will rebroadcast the message upon the timer's expiration if it does not receive any duplicate. Note that the parameter ts_d in the equation 1 is defined by b (the number of direction sectors). In this way, the farthest vehicle in each direction (sector) is assigned to the same time slot; therefore, the message is propagated in different directions simultaneously.

The main concept behind AMD is that by dividing sender's transmission range into two sectors on a highway and four sectors at an intersection, one-dimensional dissemination scheme (such as BST, DOT, etc.) is still applicable to each road direction (sector). However, such segmentation is controversial since, even on highways, road topology exists in numerous varieties such as roundabouts, curving roads, and branching roads. Also, it is probable that one directional sectors contain several road (dissemination) directions or that one road goes across more than two sectors. This leads to the concern of the suppression mechanism in AMD, where receivers make suppression decisions based on the directional sector (the number is restricted to two or four) which is defined by the road directions in the vicinity of the sender.

Consider the scenario shown in Figure 2(c), where another intersection is located in the pink sector. Since the

rebroadcast number is restricted to one in this sector, vehicles in the horizontal direction (circle in red) may wrongfully cancel their scheduled rebroadcasts once they hear the vehicle in the vertical direction rebroadcasting the message and vice versa. Once it happens, the message in this sector will be propagated in one direction only. This issue has been addressed in the literature (Liu, Yang, & Stojmenovic, 2013; Ros, Ruiz,& Stojmenovic, 2012) as the "jump over intersection" problem.

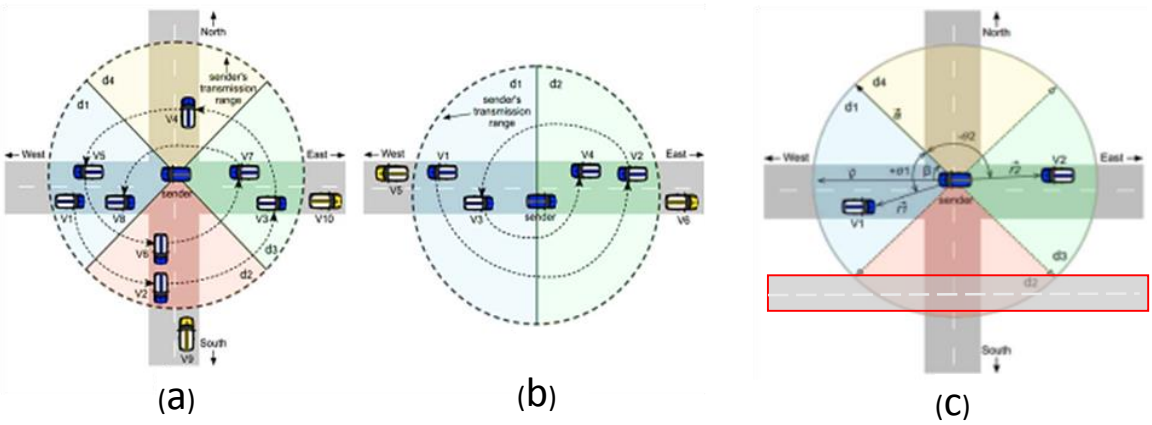


Figure 2. The multi-directional broadcasting scheme proposed in AMD for: (a) urban scenario (b) highway scenario and (c) the problem scenario of the directional sector approach.

Overall, AMD has two fundamental problems. First, dissemination directions should depend on real-time traffic directions within the vehicles' transmission range instead of the road directions in the vicinity of the sender. Second, dissemination directions numbers should be dynamically adjusted based on real-time traffic directions rather than being restricted to predetermined numbers (two or four). As a

result, AMD tends to miss some possible dissemination directions.

Chapter 3

The Protocol

This section presents a new warning message dissemination protocol called *Motion Vector Protocol* (MVP). The concept behind MVP is that due to the constraint of road topology, vehicles travelling on the same road share similar motion patterns. Each motion pattern represents a road topology as well as a potential dissemination direction. By identifying the motion pattern of one-hop neighbors, MVP enables a vehicle not only to identify potential dissemination directions without the support from infrastructure or a road map, but also to make suppression decisions without the additional information from periodic beacons.

Additionally, MVP adapted several designs to tackle the limitations of current broadcast suppression protocols. These include: adopting a sender-oriented approach to prevent hidden terminal problems; utilizing motion vector clustering schemes which help to detect real-time traffic directions and also to serve as a substitute road map; customizing cluster order sorting and CFs selection mechanism to avoid simultaneous broadcast issue which is frequently addressed in the time-slot scheme; and resolving wrongful cancellation problems with motion vector clustering and intra-cluster cancellation mechanisms.

Figure 3 shows the flow chart of the MVP, where four function modules are circled in colors. The core of MVP is the "Motion Vector Clustering" module (circled in red), where one-hop neighbors are clustered into several clusters (motion

patterns) and cluster membership information is subsequently generated to assist other modules. Under normal circumstances, MVP requires vehicles to keep executing the “Data Collection and Preparation” module (circled in blue) which gathers data through periodic beacons. Upon an event occurring, such as an accident, on-board sensors trigger MVP to execute the “Rebroadcast List Construction” module (circled in orange) which sends the warning message to one-hop neighbors through the control channel (CCH). Receivers who have been designated as a CF (by finding its Vehicle ID on the sender’s Rebroadcast List) will wait their turn to rebroadcast the message. During this waiting period, if the receiver hears a duplicate message from its neighbor, MVP executes “Intra-Cluster or global Cancellation” module (circled in green) to verify whether the scheduled rebroadcast should be cancelled or not. If the receiver hears no duplicate message upon its timer expiration, MVP also executes the “Intra-Cluster or global Cancellation” module before rebroadcasting the message. Finally, if neither Intra-Cluster Cancellation nor global Cancellation cancels the scheduled rebroadcast of the receiver, MVP executes the “Rebroadcast List Construction” module to forward the message. Details regarding each function module will be explained further in the following subsections.

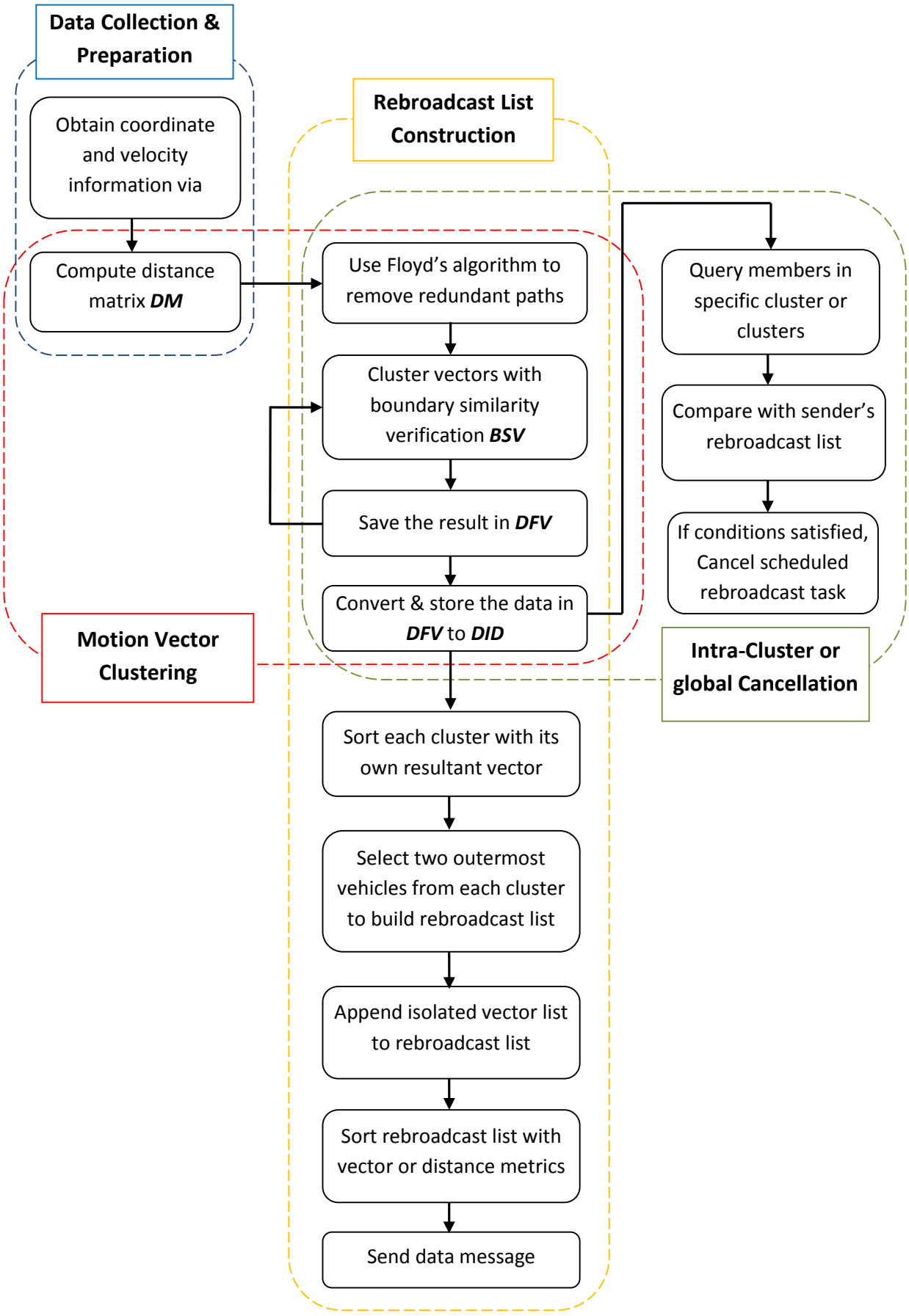


Figure 3. The flow chart of the MVP.

Requirements and Assumptions

As stated previously, neither road map nor Road Side Units are required to operate MVP. However, vehicles have to be equipped with a GPS receiver, electronic compass, and IEEE 802.11p compatible devices. Periodic beacons and data messages need to be sent through CCH in the form of WAVE Short Messages (WSMs).

The assumption here is that MVP is used or operates in the environment, where non-line-of-sight (NLOS) problem (i.e. radio signals are blocked by tall buildings) occurs less frequently. Also, the term of "curve road" often used in this work refers to simple curve, where each node of the curve shares a common center of curvature. This assumption is needed to cope with the traffic identification in the sparse network, but is optional in the dense network.

Message Structure

Two types of messages are used in this work: beacon, and data message. The beacon format is <Vehicle ID, Message ID, Time Stamp, Vehicle's Geographical Coordinates, Vehicle's velocity, Vehicle's heading>. The data message format is <Vehicle ID, Message ID, Time Stamp, Rebroadcast List, Event's Geographic Coordinates>.

To reduce the bandwidth consumption, MVP does not introduce extra information into beacons. These fields of data should already be included in Basic Safety Message (BSM) standard in U.S. or Cooperative Awareness Message (CAM) standard in EU.

As for a data message, the Rebroadcast List field contains a list of vehicles which have been recognized as CFs traveling in potential dissemination directions. Finally, a data message has a higher priority than a beacon. In this work, the priority of a data message is set to 1, whereas a beacon is set to 3.

Data Collection & Preparation

At any given moment vehicles send, collect, and prepare data for later use through periodic beacons, i.e., hello messages. Each vehicle maintains a table of received beacons (BT) which contain the latest information of one-hop neighbors and a distance matrix (DM) which store distance (dissimilarity) data for a clustering algorithm. Upon receiving a new beacon, a vehicle updates BT with new set of data and computes DM simultaneously. The Time Stamp field in beacon is used to remove outdated data from BT and DM after a predetermined interval. In the case of this research, the Interval is set to be equal to two beacon Intervals. The reason for introduction of this time tolerance mechanism is that obstacle shadowing effect or pocket collision may cause delay in message propagation as stated in the literature (Schwartz et al., 2013). Therefore, this mechanism prevents wrongful deletion of the neighboring vehicles which are not out of the transmission range.

Motion Vector Clustering Scheme

To endow the protocol with the multi-directional dissemination capability, a motion vector clustering scheme (MVC) was

proposed in this work. MVC is derived from the motion patterns study (Hu, Ali, & Shah, 2008) with necessary modification to fit this particular application.

The motion patterns study (MPS) was originally developed for a video surveillance system and was aimed at detecting motion patterns of objects in a crowded area. This unique property is applicable to a dense traffic scenario. Moreover, MPS reduces the computational overhead by adapting the motion flow field approach instead of keeping a long term motion tack of moving objects, which is particularly favorable for the application of disseminating critical messages during emergencies.

One distinctive advantage of MVC is that more vehicles can be clustered and fewer messages will be propagated. This characteristic further prevents the occurrence of broadcast storms. The efficiency and accuracy of MVC is also correspondingly improved when the density of traffic increases.

MVC considers each vehicle as a flow vector moving in a global flow field. Each vehicle is able to detect or sense a portion of the global flow field (local flow field) within its transmission range by gathering information through beacons. By clustering motion vectors of one-hop neighbors, it allows vehicles to detect a set of traffic patterns in the vicinity. Each traffic pattern (cluster) consists of a group of vehicles which are not only near one another but also participating in similar movement, which suggests that these vehicles are most likely driving on the same road. Based on this knowledge, MVP

is able to identify vehicles from different road directions and make better decisions concerning message dissemination and cancellation. Under normal circumstances, MVC only requires vehicles to keep updating their distance matrix. Once an event occurs, such as an accident, a request to send a rebroadcast list will bring up the remaining subroutines which will instantly provide the information needed. MVC functions as an on-board map and paints a topological picture of one-hop neighbors. The details of each subroutine are listed below.

a. Computing Distance Matrix

One subroutine, known as a computing distance matrix is a local field that is modeled as a weighted directed graph $G(V, A)$, where $V = \{FV_0, .., FV_{n-1}\}$ is a set of vertices representing all flow vectors (vehicles) within a receiver transmission range, and A (arc) is the set of forward distances as illustrated by $FD(i, j)$ below. In addition, a vehicle is represented by a flow vector $FV_i = (X_i, V_i)$, where X_i is the coordinate and V_i is the velocity of the vehicle.

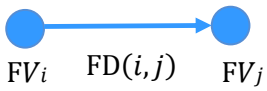


Figure 4. The proposed graph model.

In this way, each vehicle maintains an n by n distance matrix (DM), where n is the total number of vehicles within its own transmission range. Since the matrix is constructed

based on a beacon table (BT), each vehicle stores its own beacon on the top of BT and identifies itself as FV_0. Whenever content or size changes in BT, DM will be modified correspondingly, and all adjustments are triggered by a new arrival beacon.

Upon receiving a new beacon, the receiver first removes the outdated FV from BT and DM. Next, it checks whether the new arrival beacon was sent by a known sender. If so, the receiver updates FV_i in BT with new beacon by verifying vehicle ID and calculates the forward distance, i.e. $FD(0,i)$ and $FD(i,0)$, by taking X_i and V_i from FV_i against FV_0. These values, X_i and V_i, will be also computed against other vehicles' data which was previously stored in BT. Otherwise, the beacon from an unknown sender will be appended to the bottom of BT and the sender will be identified as FV_{n-1}. Following this, DM will be correspondingly expanded to the new size for storing corresponding data. Next, the receiver will calculate the forward distance in the same way as the beacon from a known sender.

The forward distance between two flow vectors i and j is defined as $FD(i,j) = \left(sd(i,j) \cdot dd(i,j)\right)^2$ also known as equation two in this work. Here $sd(i,j)$ is the spatial distance, $dd(i,j)$ is the directional difference, and taking the square of the value calculates the squared Euclidean distance.

The spatial distance is determined by the shortest distance between two flow vectors as illustrated in Figure 5(a-c). The directional difference is determined by the ratio

of the maximum upper bound angular similarity (numerator) to the prevailing angular similarity (denominator) between two flow vectors. The aim is to magnify the spatial distance between two flow vectors if they share less angular similarity.

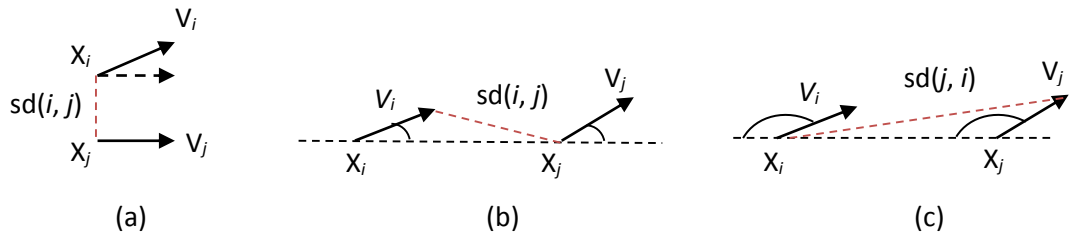


Figure 5. Spatial distance in different cases (a) vehicle i and j are on two parallel curves; (b) (c) vehicle i and j are on the same curve, where (b) i follows j and (c) j follows i.

These two values, $\mathrm{sd}(i,j)$ and $\mathrm{dd}(i,j)$, when depending on the formation of two flow vectors, are defined by the two hypotheses listed below:

1. If flow vectors i and j are on two parallel curves as shown in Figure 5(a),

   $$\mathrm{sd}(i,j) = \|\mathrm{X}i - \mathrm{X}j\|$$

   $$\mathrm{dd}(i,j) = \left(\frac{2}{1+\varepsilon+\overline{\mathrm{V}i}\cdot\overline{\mathrm{V}j}}\right)^2$$

   Where $\varepsilon = 10^{-6}$, $\overline{\mathrm{V}} = \mathrm{V}/\|\mathrm{V}\|$

   Let $\mathrm{FD}(i,j)p$ denote the forward distance between i and j in this case.

2. If they are on the same curve, and i follows j as shown in 5 (b),

   $$\mathrm{sd}(i,j) = \|\mathrm{X}i - \mathrm{X}j + \mathrm{V}i\|$$

$$\mathrm{dd}(i,j) = \left(\frac{2}{1+\varepsilon\ +\cos\theta_i}\right)\cdot\left(\frac{2}{1+\varepsilon\ +\cos\theta_j}\right)$$

$$\cos\theta_i = \overline{\mathrm{V}_i}\cdot\overline{\mathrm{X}_j-\mathrm{X}_i}$$

$$\cos\theta_j = \overline{\mathrm{V}_j}\cdot\overline{\mathrm{X}_j-\mathrm{X}_i}$$

Let $\mathrm{FD}(i,j)c$ denote the forward distance between i and j in this case.

The final forward distance (weight) between i and j is chosen as $\mathrm{FD}(i,j) = \min(\mathrm{FD}(i,j)_p\ , \mathrm{FD}(i,j)_c)$, this is known as equation 3.

Based on equation 3, the system is able to assign proper weights (forward distance) to the flow vectors in cases (a), (b), and (c)of Figure 5. Note that although the system will select $\mathrm{FD}(j,i)p$ as the forward distance in case (c) (say j follows i), an opposite scenario of i following j, this selection will not yield any wrong result because these values are still greater than $\mathrm{FD}(i,j)c$ which allows the system to select the correct (shortest) path (b) rather than (c) in the following step.

Proof: If i and j are on the same curve, in case (b), $\mathrm{FD}(i,j)_c < \mathrm{FD}(i,j)_p$. Since in hypotheses (1), $\mathrm{FD}(i,j)_p = \mathrm{FD}(j,i)_p$. Based on transitivity, in case (c), the following result can be obtained: $\mathrm{FD}(j,i)_p > \mathrm{FD}(i,j)_c$.

b. Removing Redundant Paths

In the previous step, the distance matrix (DM) is populated with appraised weights between all pairs of vertices. However, this process also introduces many redundant paths which not only leads to computational overhead but also fails to reflect a realistic topology of the traffic network.

Floyd's algorithm is capable of finding the shortest paths between all pairs of vertices within one execution. It serves a good purpose to eliminate all redundant paths by generating the shortest path matrix (SPM) from DM and then comparing DM with it. Once $\mathrm{DM}(i,j)$ is found greater than $\mathrm{SPM}(i,j)$, $\mathrm{DM}(i,j)$ is set to infinity.

c. Clustering Vectors

Traffic pattern identification is done by means of Single-linkage clustering (one of several hierarchical agglomerative clustering), where two nearby vehicles (or clusters) with similar direction are grouped into a cluster. In equation 2, the forward distance (FD) was integrated with the spatial distance and the directional difference. However, the value of FD alone does not serve as a good measure for the clustering algorithm. Due to uneven traffic distribution, it is possible that the value of FD between two vehicles in different directions is relatively smaller than that of vehicles in the same direction when the spatial distance between these two vehicles is significantly smaller, as illustrated in Figure 6. Therefore, if the clustering algorithm merely relies on the value of FD to cluster vehicles, it will result in the merging of unrelated vehicles into the same cluster. To solve this issue, the boundary similarity verification (BSV) is introduced to verify direction similarity between two vehicles (or clusters) before merging, evidenced in a later figure. In this way, the value of FD is used to identify two nearby clusters, and BSV

determines whether or not these two clusters should be merged into one cluster.

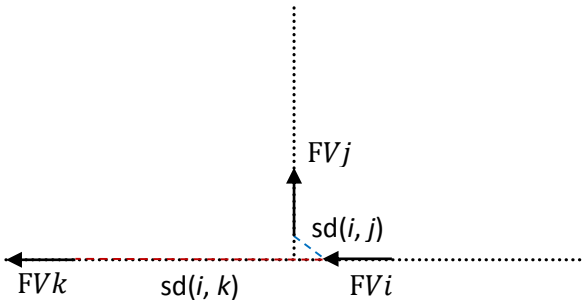


Figure 6. Illustration of problem when only relying on forward distance to cluster flow vectors

The clustering algorithm works as follows: Before clustering, the system will make an instantaneous copy of BT and DM, and then merge onto the new copy DMC so the original DM may still update data during the clustering process. In each iteration, the system first identifies two nearby vertices by searching for the minimum value (shortest arc) in DMC, and verifies them with BSV (See line 7 in Figure 7 and illustration below). If both conditions are satisfied, these two vertices (say p and q) and the minimum value (min) will be saved for later processing. (See line 5 to 13 in Figure 7). Second, merge q to p by setting the forward distance $\mathrm{D}(p,i)$ to $min(\mathrm{D}(p,i),\mathrm{D}(q,i))$ and $\mathrm{D}(i,p)$ to $min(\mathrm{D}(i,p),\mathrm{D}(i,q))$. Also, for each i, remove any arc connecting q by setting the forward distance $\mathrm{D}(q,i)$ and $\mathrm{D}(i,q)$ to infinity (See line 15 to 20 in Figure 7). Thirdly, store the result in DFV (see line 22 in Figure 7).

```
1. double min = INF;
2. double compare = -1;
3. for(int k = 0; k < oneHopNeighborNum;
   k++){
4. min = INF;
5.   for(int i = 0; i < nodeNum; i++){
6.     for(int j = 0; j < nodeNum; j++){
7.       if (i != j && min > M[i][j] &&
   M[i][j] > compare && BSV){
8.         min = M[i][j];
9.         p = i;
10.            q = j;
11.          }
12.       }
13.     }
14.     if(min == INF){ break;}
15.      for(int i = 0; i < nodeNum;
   i++){
16.          M[p][i] =
   min(M[p][i],M[q][i]);
17.          M[i][p] =
   min(M[i][p],M[i][q]);
18.          M[q][i] = INF;
19.          M[i][q] = INF;
20.      }
21.      compare = min;
22.         dataProcessing(p,q);
23.    }
```

Figure 7. The complete clustering algorithm of MVC

Finally, it keeps greedily merging until min is equal to infinity or the iteration count reaches the one-hop neighbor number (See lines 14 and 3 in Figure 7). In either case, the system will terminate the clustering process.

Boundary similarity verification (BSV)

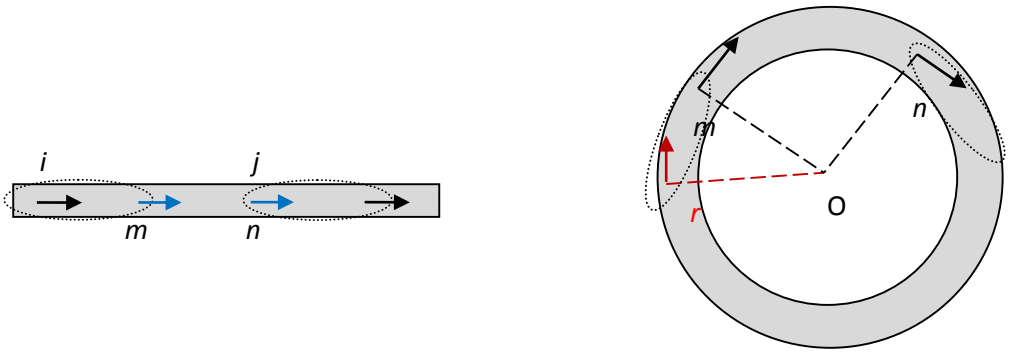
BSV is based on two hypotheses. The first is if two adjacent vehicles are traveling on the same approximate straight road, they must share a similar direction (heading). The second is if two adjacent vehicles are traveling on the same curved road, they must share a common center of curvature. To verify the existence of the curved road, a third vehicle, which shares the same center of curvature with two other neighboring vehicles, must be found.

BSV works as follows: First, upon two vertices are identified (ex. i and j) by the for loop, BSV will search DFV to locate two clusters which i and j belong to. Note that if one of these two vertices (ex. i or j) cannot be found in DFV, the vertex will be treated as a cluster which only contains the vertex itself. Second, a search for the two nearest vertices must occur, one from each cluster by referencing BTC (the copy of BT) and DFV, as illustrated by m and n in Figure 8. Thirdly, the vertices (ex. m and n) will be tested against each following scenario.

The first scenario is known as merging in line and occurs if the dot product of a velocity $\overrightarrow{Vm} \cdot \overrightarrow{Vn}$ is bigger than 0.88, and the perpendicular distance between $\overrightarrow{Vm}$ and $\overrightarrow{Vn}$ is less than

or equal to the road width. If both conditions are satisfied, the return is true.

The second scenario is known as merging on curve and occurs if m and n are on the same curve (as shown in figure 8(b)), the normal lines of their velocities will intersect at a point O and $|\overline{Om} - \overline{On}|$ will be less than or equal to the road width (hypothesis 5). If both conditions are satisfied, a third vehicle r is searched for whose velocity vector is approximately perpendicular to the vector from point O to itself $(\cos \overrightarrow{Or} \cdot \overrightarrow{Vr} \approx 0)$, in both selected clusters. Once r is found, one must test r against m and n the same way as hypothesis 5 stated above. If the existence of the third vehicle is confirmed, the return is true.



(a) Merging in Line (b) Merging on Curve

Figure 8. The illustration of (a) Merging in Line (b) Merging on curve verification in BSV

Note that if any velocity of these two vertices is equal to zero, BSV will use the heading of the vertex as the unit

vector of the velocity to conduct the verification (obtain from electronic compass and BTC). Also, the road width is used for distinguishing two similar traffic patterns from different streets. Therefore, it does not require actual size of the road width as long as the chosen road width can provide enough discriminability. In this work, the road width is set to equal one and half of the actual road width.

After applying BSV and the termination condition (line 14 in Figure 7), the predetermined number (number of clusters) in MPS is no longer needed since BSV prevents the merging between two unrelated vehicles and termination condition stops the clustering process once there is no vehicle left for merging.

d. Processing Clustering Results

Data processing is mainly responsible for two tasks: (1) saving the merged result of each iteration in an intermediate database (DFV); and (2) Converting the data in DFV into other format and then storing them to another database (DID) (see line 22 in Figure 7). Saving the merged result of each iteration in DFV is critical for two reasons. First, in each iteration, the clustering algorithm alters the data in DMC in order to find the next merging pair. Without it, there is no way knowing the membership information of each cluster.

Second, the membership information from previous iteration is required by BSV to make merging decisions. As for converting and storing the data in DFV to DID, it is designed to output the final clustering result in correct format for other modules. Throughout entire clustering process, vehicles are identified by flow vector IDs instead of vehicle IDs to

reduce computational overhead. However, the flow vector IDs are assigned by the receiver internally. In order to construct the rebroadcast list, the data needs to be first converted and then saved into DID, where the vehicle ID is used to indicate each vehicle.

Rebroadcast List Construction

In order to avoid the problems presented by distributed algorithms such as the hidden terminal problem, this work adopts the sender-oriented approach, where the rebroadcast list is generated by a sender and serves two major functions: CFs selection and the rebroadcast sequence arrangement. CFs selection consists of three steps: (a) sorting each cluster by its own resultant vector; (b) selecting the two outermost vehicles from each cluster to construct the rebroadcast list; and (c) appending the isolated vector list into the rebroadcast list. As for the rebroadcast sequence arrangement, it adapts two types of sorting algorithms. The first is based on distance and the second is based on vector as seen in Figure 9.

a. Sort Each Cluster by Its Own Resultant Vector

After executing MVC, several traffic patterns (clusters) within sender's transmission range are identified and stored in DID. However, the order of the vehicles in each cluster may be incorrect due to uneven traffic distribution affecting the merging sequence. In order to select the farthest vehicle and a serial of backup vehicles as CFs from each cluster, the correct order for each cluster must be discerned first.

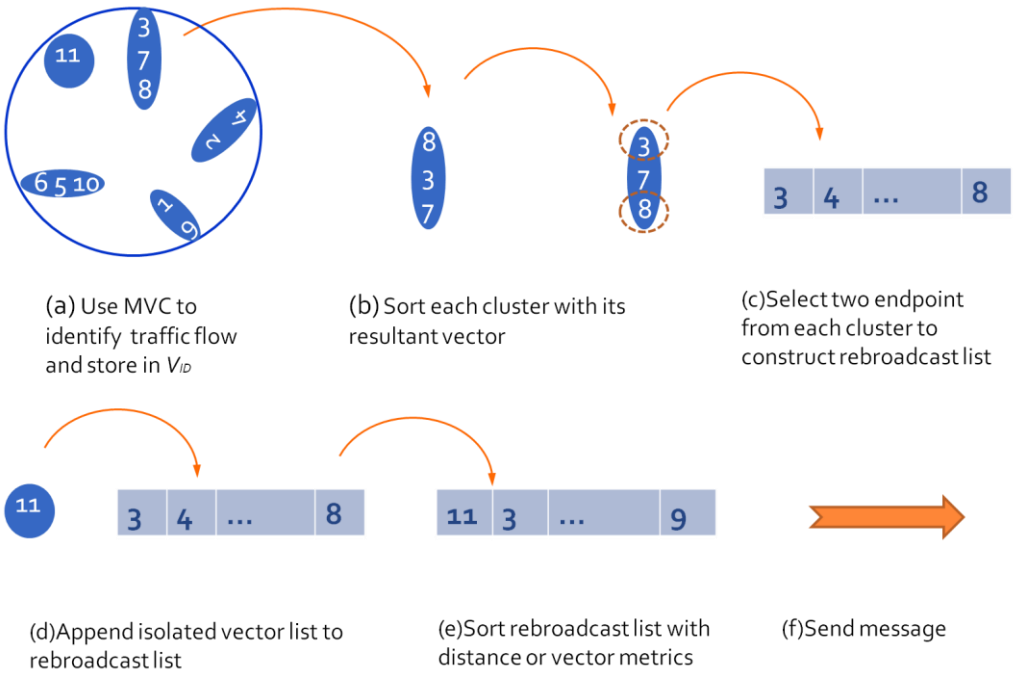


Figure 9. The Illustration of the Rebroadcast List Construction

As shown in Figure 10, the GPS coordinate system (datum) used in this work is in the WGS84 decimal degree format (Y, X), where Y represents latitude and X represents longitude; negative values of each indicates south and west, respectively. In general, there are two patterns that can be observed from the GPS reading. The first is that for a vehicle heading north, its Y reading (latitude) increases. The second is that for a vehicle heading east, its X reading (longitude) increases. The exception will be the regions which are passed by the 180th meridian. Therefore, since these regions are sparsely populated areas, they are beyond the scope of this work.

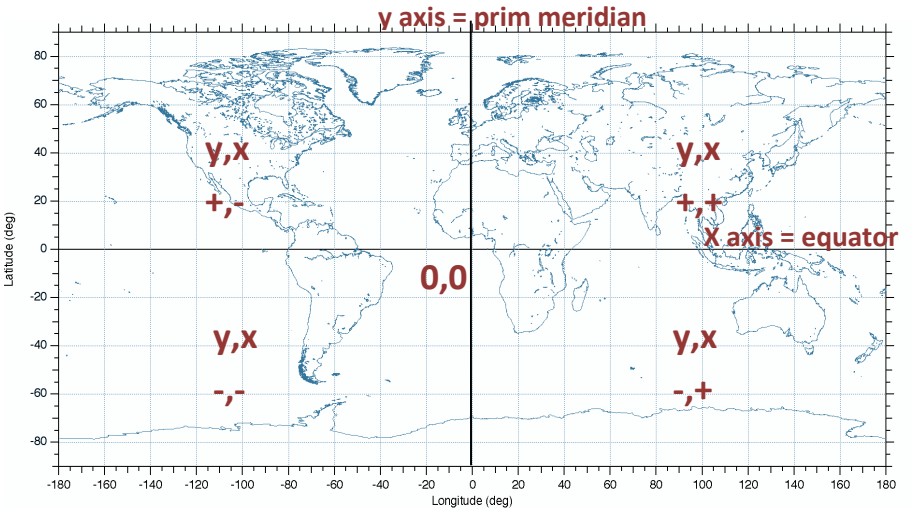


Figure 10. The GPS coordinate system (datum) used by MVP is in WGS84 decimal degree format.

Based on these two observations, it is clear that if a group of vehicles are heading straight north, their Y readings from the cluster head to the tail must be in descending order. Similarly, if a group of vehicles are heading straight east, their X readings from the cluster head to the tail must be in descending order as well. Therefore, once the general heading of the cluster is determined, the correct order of the cluster can be obtained by sorting the cluster members' coordinates in ascending or descending orders.

The general heading of a cluster can be acquired by summing the velocities (or headings) of each vehicle within the cluster. In other words, the resultant vector of entire members' motion vectors in a cluster can be used to determine the general heading of the cluster. In this way, sorting cluster members by the cluster's resultant vector, gives an exact order of the vehicles in that cluster (road direction),

which enables the algorithm to (1) identify the farthest vehicle in each cluster, (2) identify a serial of backup candidates nearby the farthest vehicle for taking over the rebroadcast task once the vehicle with a higher priority fails to rebroadcast, and (3) select CFs linearly regardless of the road shape and width, which avoids the common segmenting problems and difficulty in the time slot scheme.

b. Select Two Outermost Vehicles from Each Cluster to Construct the Rebroadcast List

After all the clusters in DID have been sorted, the farthest vehicles in each cluster can be systematically selected. In this work, the farthest vehicles in a cluster are defined as the two outermost vehicles of the cluster. For added reliability in case a CF fails to rebroadcast, the algorithm will iteratively select the next two outermost vehicles from outside inward as backup candidates until it reaches a predetermined number CF(i). The candidate forwarder number CF(i), is the number of vehicles chosen to be a CF from one cluster, where i is an even number {2,4,6,…}. For instance, CF4 represents selecting two CFs from either side of the cluster.

The decision to select two outermost vehicles from each cluster is based on the fact that a one-way street is a common layout in metropolitan areas. Since vehicles on a one-way street might be the only media onsite to forward messages, it is essential to designate vehicles from both endpoints of a cluster as CFs in a case where there is no traffic in the opposite direction and the opposite direction street might be

located one or more blocks away. Although it potentially increases the rebroadcast redundancy while having traffic in both directions, without this mechanism, messages might not be disseminated to all possible directions. More conventional distance based approaches suggest selecting one vehicle that is farthest from the sender in the traffic direction, which results in propagating the message only in one direction (Wisitpongphan, et al., 2007; Suriyapaibonwattana et al., 2008) as shown in Figure 11(a,b). Therefore, in a 2-dimensional (urban) scenario, both outermost vehicles should be considered as the farthest vehicles in one particular traffic direction (cluster) regardless of whether their distance to the sender is the farthest or not, as illustrated in Figure 11(c).

c. Append Isolated Vector List to Rebroadcast List

In graph theory, an isolated node by definition is a node that is not an endpoint of any edge. In the case of this research, an isolated vector is the only member in a cluster without any neighbor sharing the same motion pattern as it does, as illustrated in Figure 12.

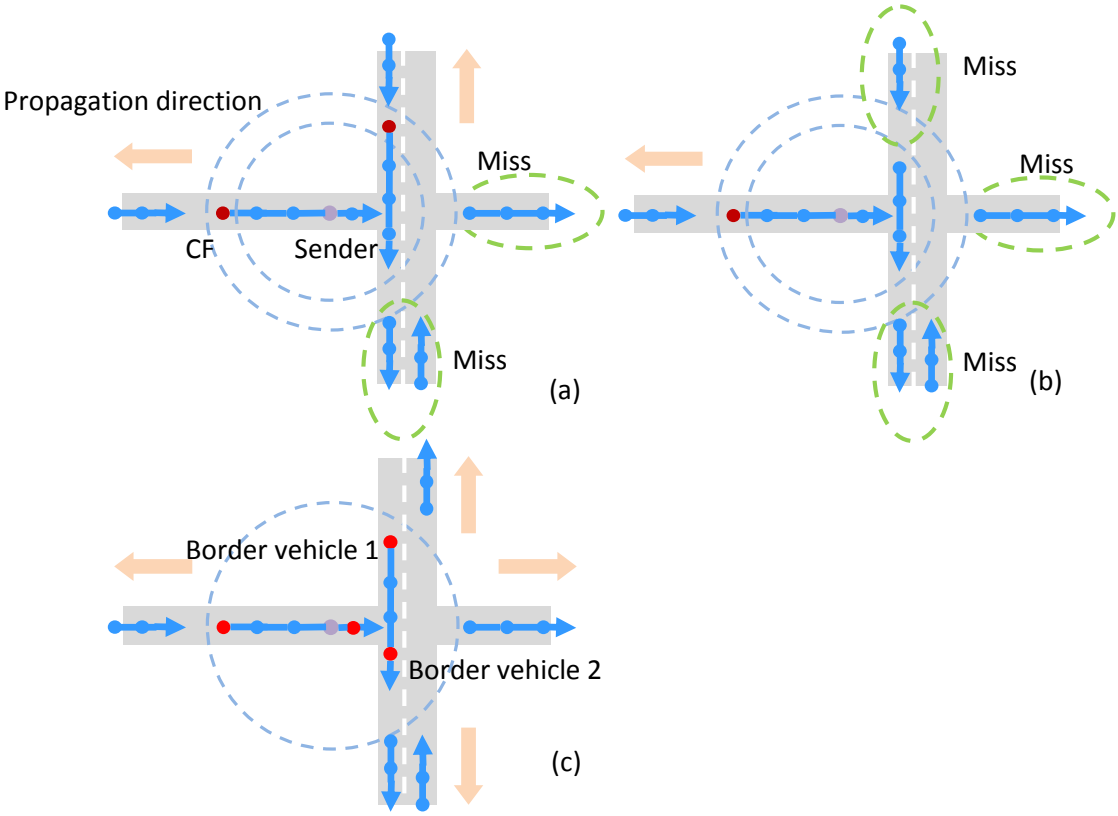


Figure 11. Propagation problems occur when one relies on (a) distance matrix (b) as the only one message forwarder. The proposed scheme is shown in (c).

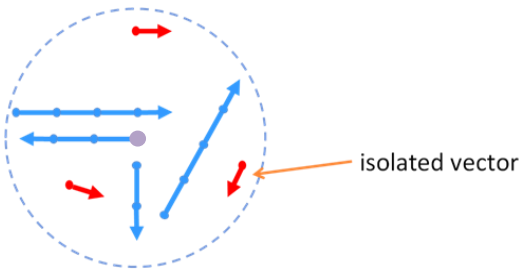


Figure 12. The illustration of an isolated vector.

The decision to include these vehicles into a rebroadcast list is that in low density networks, the isolated vehicles clearly help to increase the reachability of the protocol. Also, on some rare occasions, these vehicles happen to be the

only gateway nodes connecting two or more separated networks. This tends to increase the rebroadcast redundancy and latency of the protocol, but when developing such safety related protocols, this is a trade-off that needs to be accepted. Additionally, the isolated vector list can be obtained by excluding members in DID from BTC.

d. Sort Rebroadcast List with Vector or Distance Metrics

In order to avoid message collisions and contentions, the sender will arrange the rebroadcast sequence (list) before transmitting the message. The rebroadcast list is used to facilitate a time delay mechanism which creates a linear time delay sequence among CFs so that they will rebroadcast their messages one at a time. This separation in time not only allows the message to be quickly disseminated by the vehicles with a higher priority but also provides the time for those vehicles with lower priority to cancel their rebroadcast task when a duplicate message is received from one of its neighbors.

In this work, two types of senders are defined: the original sender and message forwarding sender. The original sender is the vehicle directly involved in an event such as an accident and the generating of an initial message regarding the event. The message forwarding sender is the vehicle receiving the message and forwarding it subsequently. For the sake of fast propagation, two types of sorting algorithms are adapted.

The first type is the distance matrix. This algorithm sorts the list based on the distance between the sender and the CFs

in descending order (i.e. with farther vehicles on top of the list) which follows a spiral shape pattern as shown in Figure 13(a) (Schwartz et al., 2013). The second type is known as a vector matrix. This algorithm sorts the list according to the propagation direction which is defined by the vector linking the previous sender to the current sender as shown in Figure 13(b), where the CFs are not covered by previous broadcast are put on top of the list.

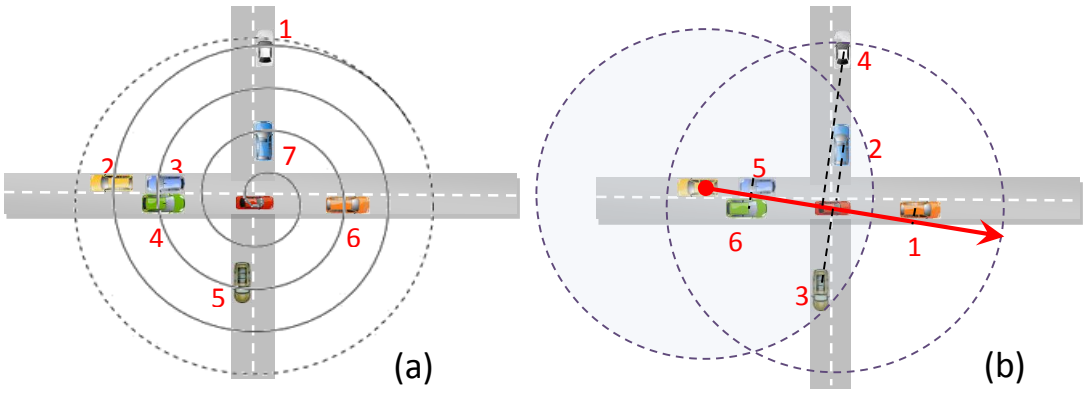


Figure 13. The illustration of (a) a distance matrix and (b) a vector matrix.

In general, the distance matrix can achieve fast propagation and low rebroadcast redundancy since it always assigns higher priority to the CFs around the periphery of a sender's transmission range. These CFs have higher potential to reach other vehicles located outside the range. Once an outer CF of a cluster has successfully rebroadcasted the message, the inner CFs in the same cluster cancel the scheduled rebroadcasts correspondingly. Therefore, it can significantly

reduce the number of transmissions (rebroadcast). It works particularly well on multi-directional broadcasting, which disseminates messages to the complex road topology such as urban streets. Due to this, the original sender sorts the list based on this matrix. As for the message forward sender, it can adapt either of the two sorting algorithms depending on which type of road topology it is in.

However, the distance matrix might not be the best way to sort the list for a message forwarding sender. As shown in Figure 13(b), a message forwarding sender shares an overlapping coverage area (OCA) with the previous sender. According to Tseng et al. (1999), the percentage of OCA over the current sender's transmission coverage area ranges from 39-100% and 59% on average. If the distance matrix is used in this case, it is probable that appreciable amounts of vehicles within the OCA will be given high priority to rebroadcast.

Since the majority of vehicles within the OCA have received the message from the previous sender, it may result in increasing the broadcast redundancy (if the system allows these vehicles to rebroadcast freely) or the latency of the dissemination (if the system suppresses the rebroadcasts in this area, the CFs outside the OCA have to wait for their turns). In the latter case, depending on the density and location of the vehicles within the OCA, the latency of the distance matrix is unpredictable. One conventional solution is that every vehicle attaches the last received message as acknowledgment (ACK) into its beacons to notify the neighboring vehicles that the message has been properly

received (Ros et al., 2012). In this way, the message forwarding sender can use the information acquired from beacons to exclude the vehicles within the OCA from its rebroadcast list. However, considering the time span between beacon (0.5 s) and the duration of entire message dissemination (0.1-0.3 s), it is impractical to adapt this approach.

An alternative solution is to use the vector matrix to sort the list for a message forwarding sender. The simulation results indicate that sorting the rebroadcast list in this way not only shortens the wait time for the CFs outside the OCA but also has better coverage than the distance matrix. Essentially, the vector matrix is suitable for unidirectional broadcasting, which disseminates messages to monotone road topology such as highways or interstates. Overall, the distance matrix suppresses more redundant rebroadcasts, whereas the vector matrix has better dissemination coverage. Therefore, this work mainly focuses on the vector matrix.

The above scenario can be better illustrated by the following example. Consider a second scenario(as shown in Figure 13(b)), where the red car receives a message from the yellow car and is designated to forward the message. If the red car’s rebroadcast list was sorted by the former sorting algorithm (distance metrics, as shown in Figure 13(a)), the yellow car would be assigned to a higher priority (position 2) than the orange car would be assigned to (position 6) even though the message was sent by the yellow car. Due to the suppression mechanisms (which will be elaborated further in

the following section) implemented in this work, all of the cars within the yellow car's transmission range have already heard the message and will simply discard the duplicate message sent by the red car. The orange car is then unnecessarily forced into a long waiting period, before finally transmitting the message.

Delay-based suppression scheme

Figure 14 shows the algorithm of the broadcast suppression scheme, which works as follows: Upon receiving a message, the receiver checks whether the message or its duplicates have been received previously by comparing the message ID with known message IDs. This verification is achieved by maintaining a table of received message (MT) which contains the copies of updated first-time received messages. If a message is received for the first time, a copy of this message will be stored and remain in MT for an interval of time which is prescribed by the sender in the time stamp field of the message. Once the lifespan of the message has elapsed, the system will remove the copy from MT.

After the message has been confirmed as a first-time received message, the receiver checks the rebroadcast list, which was attached to the message, to see if it has been designated as a CF. If the receiver is designated to be a CF, it will setup a delay timer based on the position of its vehicle ID located on the list; otherwise it will simply discard the message.

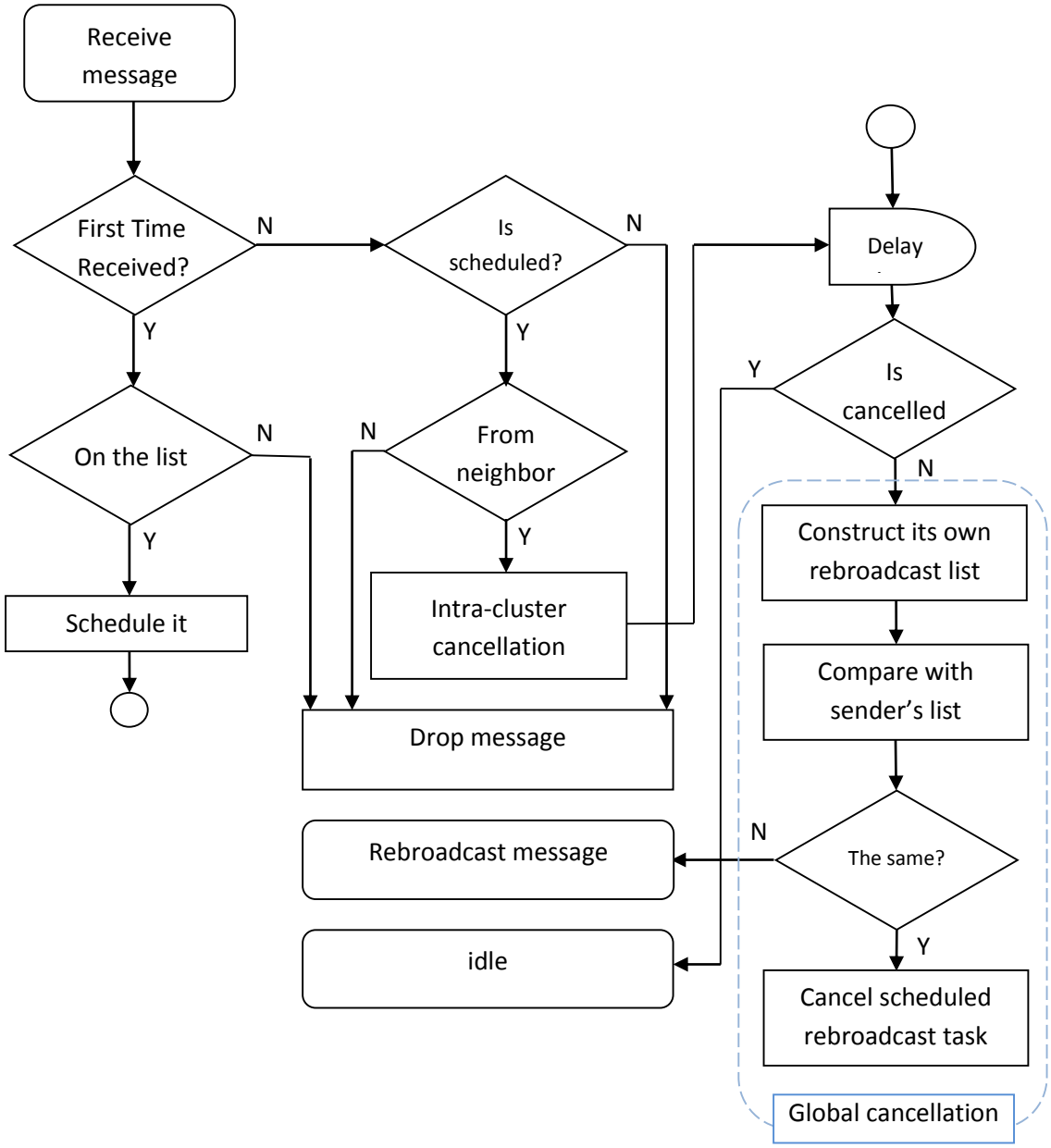


Figure 14. The delay-based suppression scheme used by MVP.

The delay timer is defined as: $\mathrm{Tdelay}(i) = i \times \tau$ (Li et al., 2013. Where i is the receiver's position on the rebroadcast list, and τ is the estimated one-hop delay. Once the delay timer has been scheduled, the receiver will wait for its expiration. During that time, if the receiver receives any duplicates of the message, it will check whether the duplicate was sent by one of its neighbors. If the sender is one of its neighbors, the receiver will initiate an intra-cluster cancelation; otherwise, it will ignore the message and keep waiting on the timer.

Intra-cluster cancelation is where the receiver decides if the rebroadcast task should be cancelled by its neighbor or not. Note that the neighbor referred here is not only geographically adjacent to the receiver, but also traveling with the receiver on the same road. More precisely, the neighbor is one of CFs that the sender designated as a CF along with the receiver in the rebroadcast list for that specific traffic pattern (cluster). Since the sender always selects the CFs from both endpoints of a road segment (cluster), the receiver needs to verify where the duplicate originated from. By default, the receiver only cancels the rebroadcast task when it hears that the neighbor on its own side of the cluster has rebroadcasted the message. As stated previously, this allows the message to propagate in two directions (per cluster). If the intra-cluster cancelation does not cancel the rebroadcast task, after the timer expired, the receiver will initiate a global cancelation. More details

regarding intra-cluster and global cancelation will be explained further in the following section.

As elaborated above, vehicles only handle first-time received messages and rebroadcast the same message once. There are three reasons for this restriction. Firstly, according to the simulation results, normally it takes less than a hundredth of a millisecond for the vehicle located 3.5Km away to receive the message. Repeating rebroadcasts from the same vehicle will not increase additional coverage since the position of vehicles barely change during such a short interval. Secondly, it can prevent the broadcast loop between sender and the receiver from happening. Thirdly, the vehicles within overlapping coverage areas from different senders' transmission ranges will not be forced to reschedule the same message over and over again.

Intra-Cluster and Global Cancellation

Figure 15 shows the complete algorithm of the intra-cluster cancellation scheme, which is achieved by providing the cancellation lists for the vehicles that have been waiting on their delay timers. The whole process is triggered by the first duplicate message which may be sent by the vehicle in different directions. The first task for the receiver is to identify whether the duplicate message was sent by its own neighbors or not. In order to do that, the receiver must first acquire the updated cluster membership information, i.e. DID, by executing MVC. Second, the receiver identifies which cluster itself belongs to and saves all members within the cluster to the members list ML.

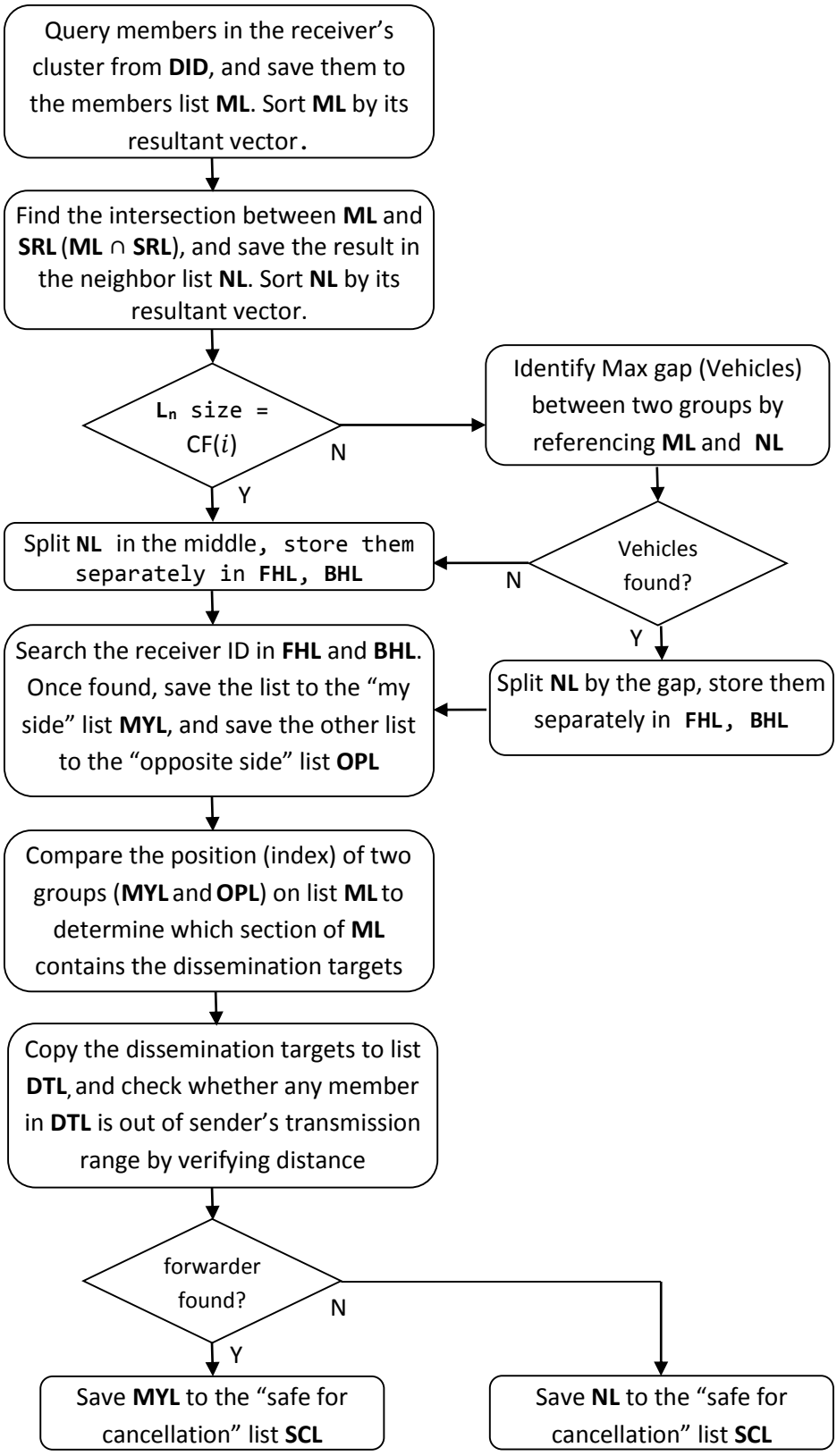


Figure 15. The complete algorithm of the intra-cluster cancellation scheme.

Thirdly, the sorting of ML by its own resultant vector to attain the correct order must occur. Finally, by finding the intersection between ML and SRL (sender's rebroadcast list), the neighbor list NL is obtained.

This list contains CFs who were chosen from both sides of the cluster (two groups) based on the sender's perspective. However, to reduce transmission overhead and channel congestion, in our design, the sender does not provide any information regarding which side the receiver was chosen from and how these two groups were divided. Therefore, the receiver must rely on local knowledge (attained from beacon) to identify the neighbors on its own side of the cluster (its own group) for cancellation. To achieve this goal, three strategies were used in this work; they are: splitting list, finding a maximum gap, and identifying dissemination targets.

Splitting list NL in the middle is applied when the receiver is able to sense all designated CFs in its traffic direction (cluster). In this case, the size of NL is used to identify this condition. When the size of NL is equal to the system designated CF size CF(i), the receiver will first split the sorted NL in the middle, and store these two sub lists separately in FHL (front half of the list)and BHL(back half of the list). Then, it will identify which group it belongs to by searching its own ID in FHL and BHL. Once found, it will save the list containing its ID to the neighbors on my side list MYL, and save the other list to the neighbors on opposite side list OPL.

Although this splitting works flawlessly in this case, due to obstacle shadowing effect or coverage difference, the number of CFs a receiver is able to detect might be less than $CF(i)$. In such a scenario, equally dividing the list in half may result in assigning some backup CFs into wrong groups and causing these CFs waiting on the sender on opposite the side to cancel their scheduled rebroadcasts. As a result, the broadcast redundancy increases. Therefore, finding the maximum gap strategy is introduced to split the list properly.

The mechanism works based on the design that the sender always selects outermost vehicles from a cluster sequentially. By identifying the traffic between two groups of CFs, the algorithm is allowed to distinguish two groups and subsequently separate them. It works as follows: First, it identifies and records the position (index) of known neighbors (CFs) on the list ML by referencing NL. Second, it searches the maximum position difference among CFs and records the CFs' IDs who share the maximum variation. Third, it uses the IDs as a reference to split NL into FHL and BHL. Finally, one must identify MYL and OPL as stated above.

The reason for using position as metrics instead of distance to conduct this identification is that the distances between CFs might be greater than the target gap (with traffic in it) due to random traffic distribution. Similarly, more than one gap may exist among CFs due to the high mobility nature of traffic or obstacle shadowing effect. Therefore, searching the maximum variation ensures finding the exact gap where two groups are separated. This method is favorable

because it works particularly well for dense traffic segments (clusters), where the broadcast storm most likely occurs.

In contrast, for a sparse traffic segment, the focus will be on disseminating critical messages to all possible directions rather than reducing the broadcast redundancy. One indication of such scenario is that the algorithm found no traffic (gap) among CFs, therefore, it splits list NL in the middle to ensure that the message will be propagated in two directions (per cluster). As stated above, splitting list in this way potentially increases broadcast redundancy, but the following method helps to compensate for the effect.

The third strategy enables the receiver to select the proper cancellation list by identifying dissemination targets in the designated direction. After the previous step, the receiver is classified into one group which is responsible for propagating messages to a designated direction. It also obtains two cancellation lists. Among them, list MYL contains the neighbors from its own group, and list NL may contain the neighbors from both groups. The selection criterion between these two cancellation lists is based on whether or not the receiver could discover any forwarder who is outside a sender's transmission range in the designated direction. If it is true, list MYL is chosen to be the cancellation list Ls to prevent wrongful cancellation by the other group. Otherwise, since the scheduled rebroadcast is redundant, cross group cancellation is allowed by selecting NL as the cancellation list. This mechanism is achieved by identifying the

dissemination targets, which are in the designated direction, on the members list ML.

Comparing the position (index) of two groups (MYL and OPL) on list ML allows the algorithm to determine which section of the list contains the dissemination targets. Once found, the entire membership of the section will be saved to the dissemination targets list DTL. After that, the distance between the sender and each member in DTL will be examined individually, and the final result is used to select a cancellation list as stated above.

Note that, without regard to the designated rebroadcast sequence, this method suppresses the redundant messages in two ways. In one way, for a CF that is classified into a correct group, it verifies the necessity of rebroadcasting before actually executing it. In another way, if a CF is misclassified into wrong group, the CF will not only suppress its rebroadcast but also select NL as the cancellation list, (i.e. the CF back to correct group). The reason is that if a CF is misclassified into wrong group, the designated direction of the CF will point to a wrong direction such as pointing inward instead of outward from the sender. As a result, the entire members in DTL are most likely within a sender's transmission range causing the CF found to have no forwarder in its designated direction. This property indirectly resolves the splitting dilemma which was introduced previously.

Global cancellation, on the other hand, aims to verify whether or not the receiver shares the same rebroadcast list with the sender before forwarding the message. The receiver

will first construct its own rebroadcast list and find the difference between two rebroadcast lists. If they are the same, the scheduled rebroadcast will be cancelled, since it will not increase any additional coverage. Additionally, to reduce computational overhead, instead of directly executing global cancellation upon receiving a message, the algorithm only executes it when intra-cluster cancellation does not occur.

Chapter 4

Simulation Results

To better examine the capability of the proposed model, a real map fragment is particularly selected from Dammam, Saudi Arabia, consisting of a roundabout and various type of road sharps. It was acquired from OpenStreetMaps (OpenStreetMap Contributors, 2015), and has an area of 3Km X 2.5Km as shown in Figure 16.

Figure 16. The map fragment of Dammam, Saudi Arabia.

Ten random realistic traffic patterns were generated by Sumo 0.19.0 (Krajzewicz et al., 2012) with five different node densities ranging from 100 to 300 nodes, in total 400 were run for this scenario. The simulations were run on a discrete event simulator, OMNET++ 4.6 (Varga, 2015) which was connecting with Sumo by Veins 2.2 framework (Sommer, German, &

Dressler, 2015). To simulate real word radio propagation environment, two-ray ground path loss model and simple obstacle shadowing model were also applied in this experiment. In each simulation, an accident site is randomly selected, and the first vehicle encounters such event generating the initial warning message. The rest simulation parameters are listed in Table 1.

Table 1. Simulation parameters.

Phy80211p	Frequency band	5.89GHz
	Max TXPower	10mW
	Sensitivity	-89dBm
	Thermal Noise	-110dBm
Mac1609.4	TX Power	20mW
	Bit Rate	18Mbps
WaveAppLayer	Header Length	256 bit
	Beacon Interval	0.5s
	Max Offset	0.005s
Simulation time		30s

The following metrics proposed by Tseng et al. (1999) are used to evaluate the results. The first is reachability (RE) and is the number of vehicles receiving the broadcast message divided by the total number of vehicles that are reachable, directly or indirectly, from the source host. The second is saved rebroadcast (SRB) and is (r - t)/r, where r is the number of vehicles receiving the broadcast message, and t is the number of vehicles actually broadcasted the message. Lastly is what is known as average latency. This is the

interval from the time the broadcast was initiated to the time the last vehicle finishing its rebroadcasting.

In the following sections, simple flooding (each node broadcast once) was used as a baseline for comparison. Figure 17 examined the total number of transmissions. Using flooding, the number of transmissions increased linearly as the number of nodes increased. Using CF2 and CF2 without isolating vectors, the numbers of transmissions were significantly reduced.

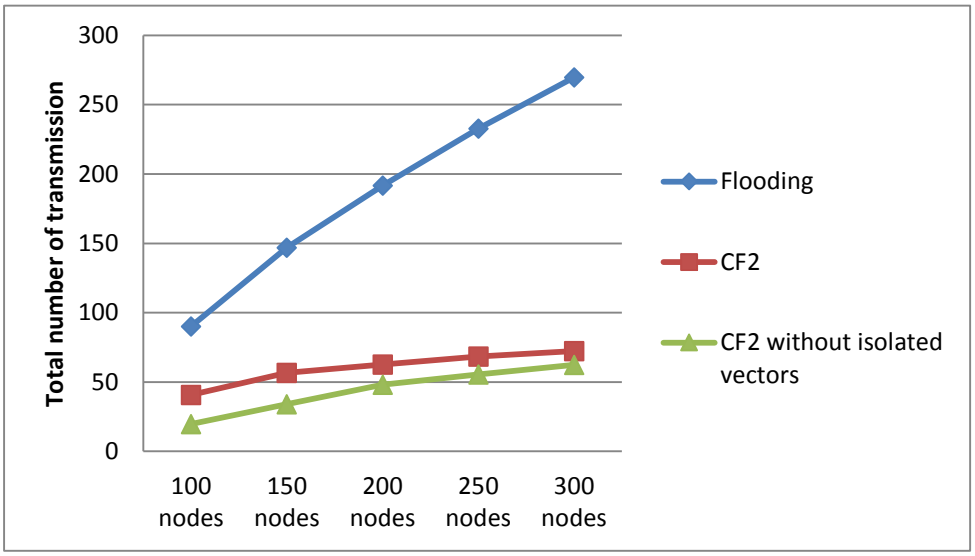


Figure 17. Total number of transmission (rebroadcast) in different node density based on vector sorting.

Figure 18 examined the number of saved rebroadcasts. Because flooding causes all nodes to rebroadcast, there was zero numbers of saved rebroadcasts. When using CF2 and CF2 without isolating vectors a saved rebroadcast percentage of 73% and 77% respectively were observed. While both were

significantly better than flooding, there is a statistically insignificant difference between using CF2 and CF2 without isolating vectors.

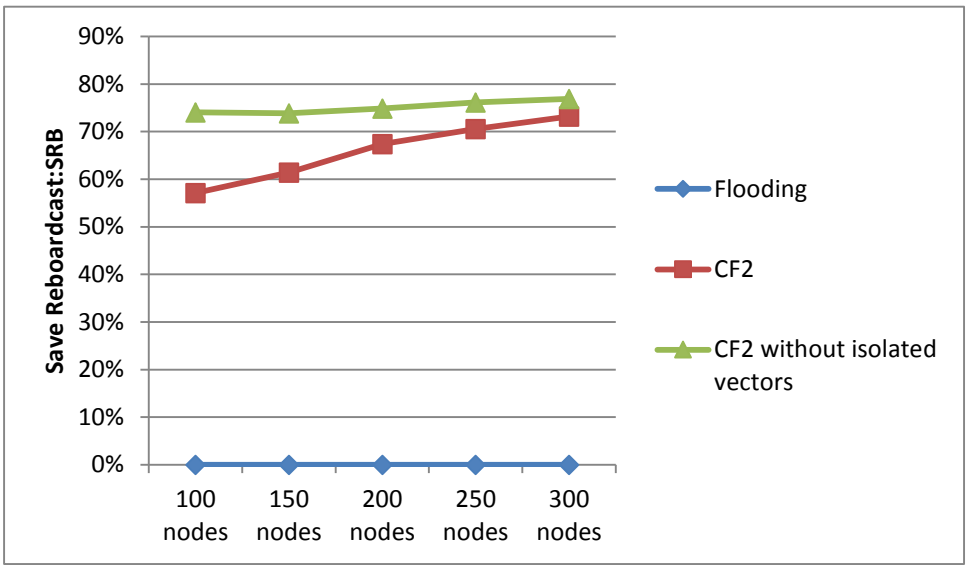


Figure 18. Save rebroadcasts in different node density based on vector sorting.

Next Figure 19 demonstrates reachability. Because flooding has every node broadcast the message, eventually 100% of nodes will have been reached. When using CF2 and CF2 without isolating vectors, CF2 immediately reached 100% coverage. CF2 without isolating vectors, reachability was only about 80% in low density traffic (<200 nodes). In higher density traffic (>= 200 nodes) the reachability once again reached 100%. It is suspected that in lower density traffic the probability of having a vehicle that is not clustered with other vehicles (an isolating vector) is increased as there are fewer cars on the road. Without utilizing these isolating

vectors there are potential coverage gaps which may mean that some vehicles will be unable to receive the message. This is less of a problem in higher density traffic, as the sheer number of vehicles will allow for 100% coverage.

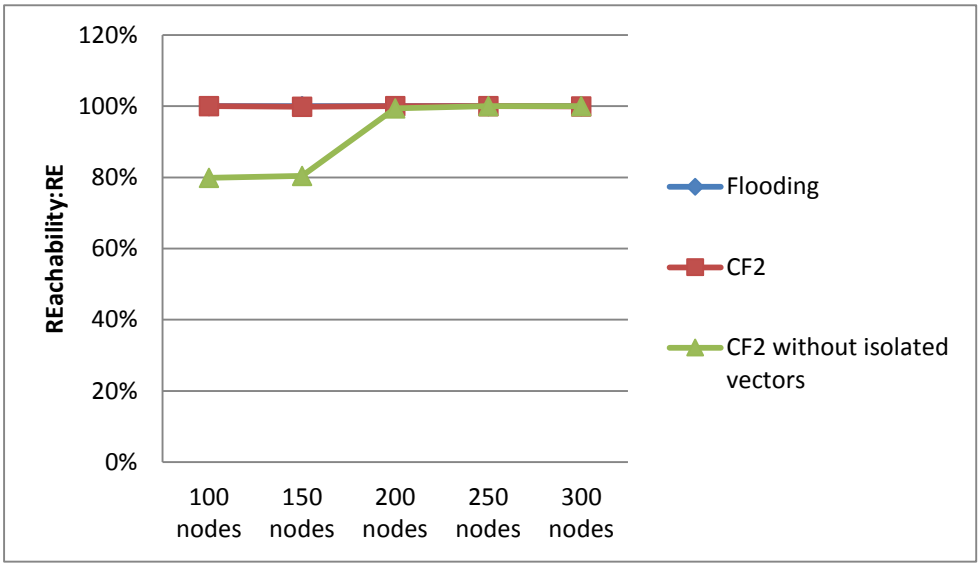


Figure 19. Reachability in different node density based on vector sorting.

Finally, the latency can be examined in Figure 20. Again, because flooding forces all nodes to rebroadcast, it can propagate its message to all nodes very quickly. In CF2 and CF2 without isolating vectors the time needed to reach every node was greater than flooding. This is because CFs must wait for their delay timers to expire before broadcasting. Again, in higher density traffic the differences between CF2 and CF2 without isolating vectors were diminished.

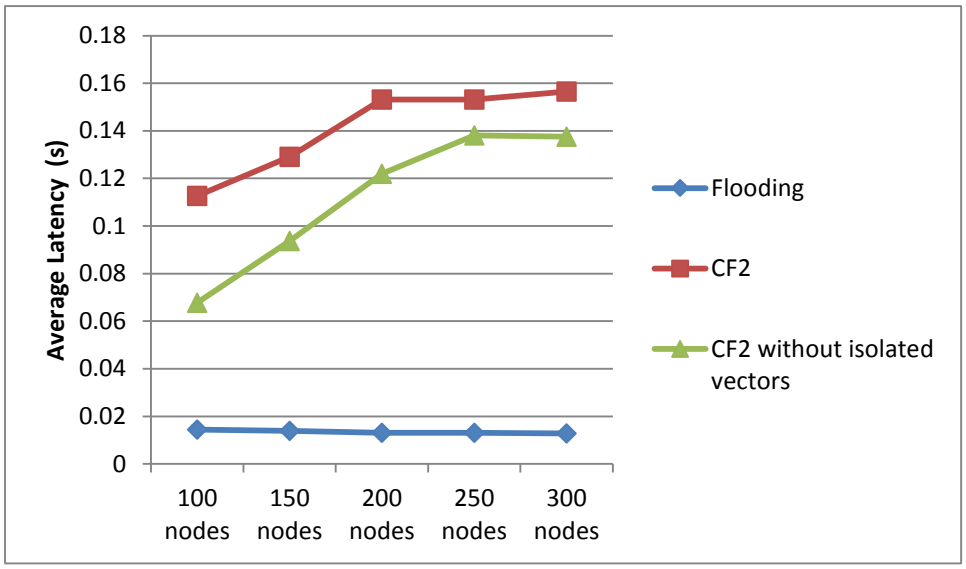


Figure 20. Average latency in different node density based on vector sorting.

Overall, the simulation results justify the decision to include the isolating vectors (vehicles) into a rebroadcast list. Although it slightly increases the rebroadcast redundancy, it clearly helps to achieve high coverage in the sparse network.

In the following section, CF2, CF4, and CF6 are compared and contrasted. Figure 21 compares the number of total broadcasts between the three models, showing a statistically insignificant difference in the number of total rebroadcasts. This indicates that message cancelation strategies are working as expected.

While higher CF(i) will help ensure no message are lost, they will also slightly add to the latency. There are three possible causes of the latency. First, the vector matrix (sorting algorithm) provides non-optimal rebroadcast sequencing for some clusters causing the CFs around the periphery of a sender's transmission range to have to wait

extra time before transmitting the message. Second, some CFs with a high priority may be located inside the OVA; therefore, in this scenario the rebroadcast is done by the CFs with a low priority. Finally, increasing the number of backup CFs results in a larger packet size, which may increase the chances of message collision. Careful consideration should be taken when considering adopting a higher CF(i).

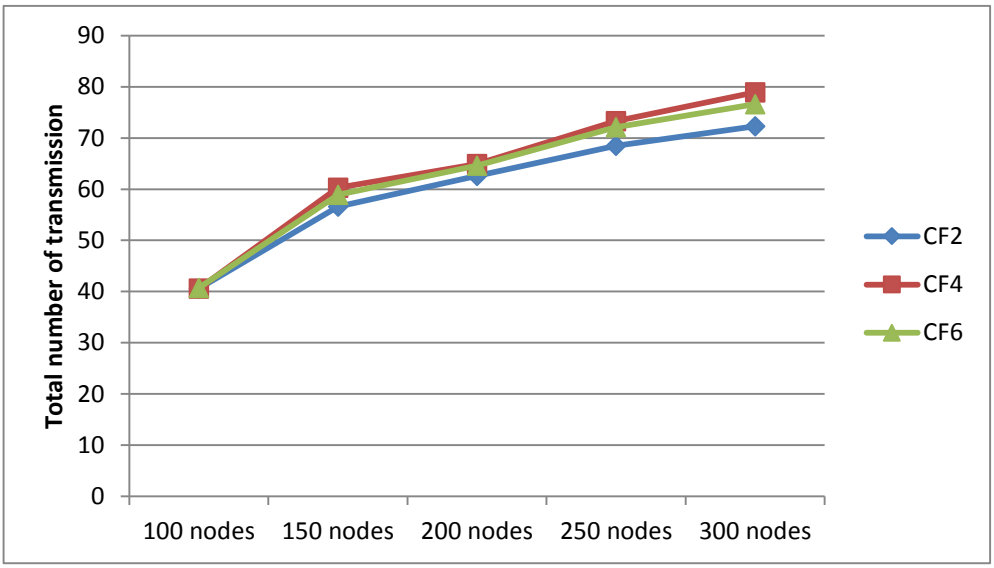


Figure 21. The comparison of total number of transmission by varying candidate forwarder number CF(i) based on vector sorting.

Finally, simulation results of the distance matrix are shown in Figure 23 and 24. By Comparing these results with that of the vector matrix (in Figure 21 and 22), it is evident that the distance matrix yields lesser transmission but longer latency than the vector matrix.

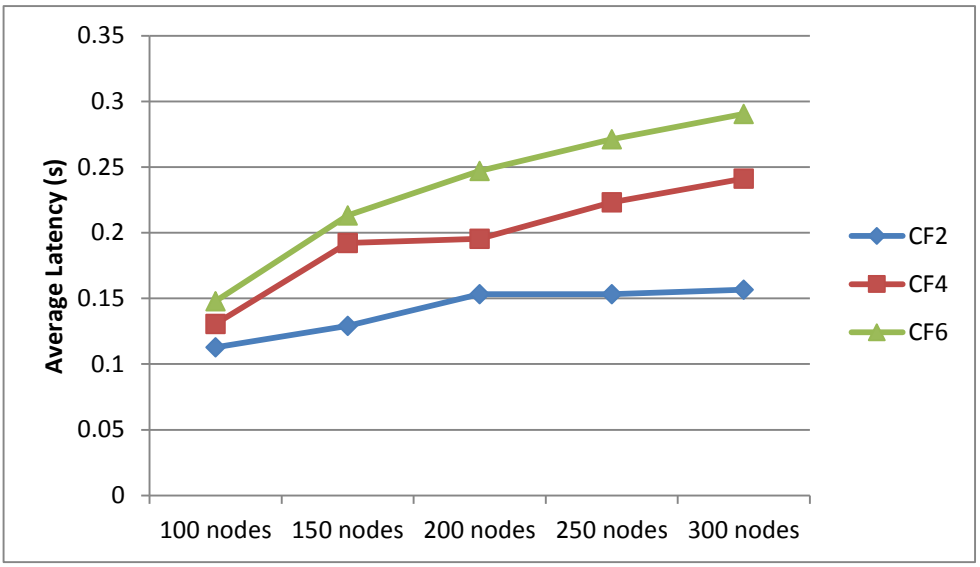


Figure 22. The comparison of average latency by varying candidate forwarder number CF(i) based on vector sorting.

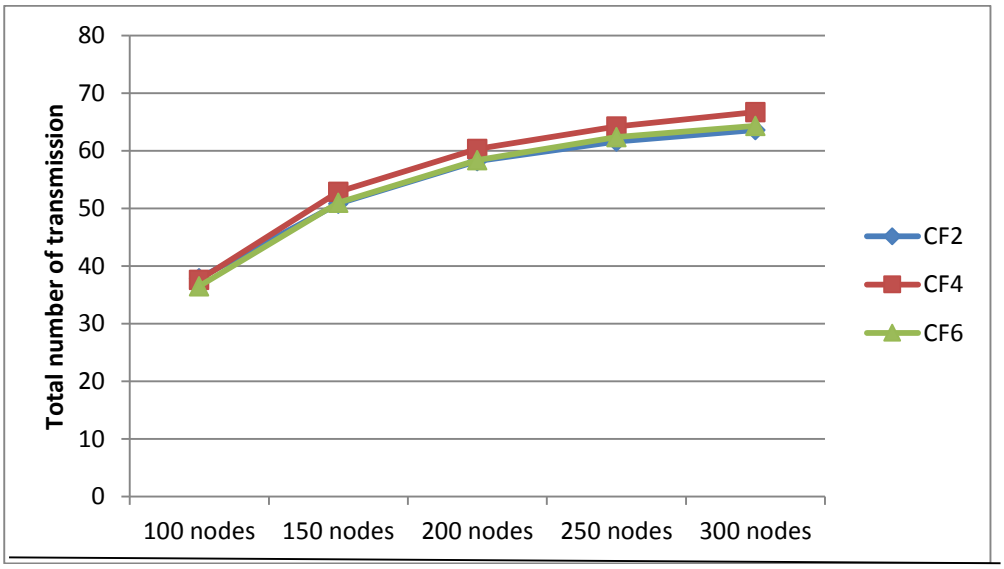


Figure 23. The comparison of total number of transmission by varying candidate forwarder number CF(i) based on distance sorting.

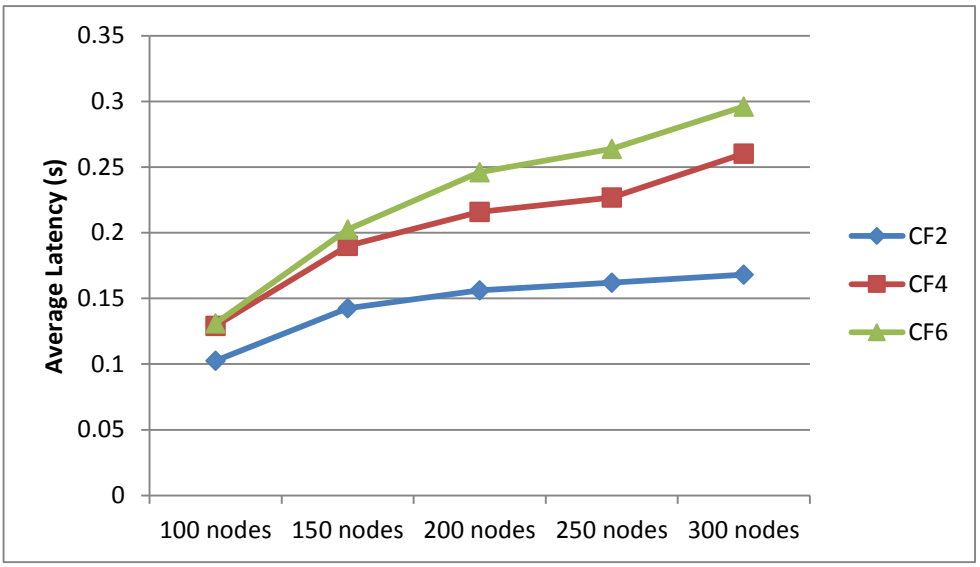


Figure 24. The comparison of average latency by varying candidate forwarder number CF(i) based on distance sorting.

The main concern of the distance matrix is that in some rare occasions, the distance matrix fails to achieve full coverage (as shown in Figure 25), unlike the vector matrix. Since this work mainly focuses on delivering safety related messages, the reachability should be valued higher than the broadcast redundancy. As a result, this work adopts the vector matrix as the primary sorting algorithm for arranging rebroadcast sequence. Further study is needed to address the coverage issue occurring in the distance matrix.

Figure 25. Reachability issue when adopting distance sorting.

Chapter 5

Conclusion

Most broadcast suppression protocols mainly focus on one-dimensional message dissemination model. Only a few protocols address the need of multi-directional message dissemination scheme in VANET context. This research provides a new protocol (MVP) with the ability to distinguish real broadcast directions which status quo protocols lack. The experimental results clearly show that the proposed MVP protocol is workable and useful. Also, unlike current segmenting approaches which rely on unrealistic assumptions of network topologies, MVP protocol captures the network topology by means of motion vector clustering which enables it to operate on complex road topology and also identify dissemination directions in the moment.

Future work will focus on optimizing the performance of MVP. Currently, the MVP protocol only allows CFs to rebroadcast their messages one at a time. By utilizing cluster membership information and analyzing coordinate differences among CFs, it is possible that multiple CFs will be able to rebroadcast simultaneously without causing any broadcast storm issues. Another possible direction will be to focus on implementing a store-and-carry mechanism to cope with the disconnected network issue.

Bibliography

Hu, M., Ali, S., & Shah, M. (2008). Learning motion patterns in crowded scenes using motion flow field. *Proceedings of the 19th International Conference on Pattern Recognition (ICPR 2008)* (pp. 1–5). Tampa, FL: IEEE.

Korkmaz, G., Ekici, E., & Özgüner, F. (2006, June). An efficient fully ad-hoc multi-hop broadcast protocol for inter-vehicular communication systems. *Proceedings of the IEEE International Conference on Communications (ICC 2006). 1*, pp. 423 - 428. Istanbul, Turkey: IEEE.

Korkmaz, G., Ekici, E., Özgüner, F., & Özgüner, Ü. (2004, October). Urban multi-hop broadcast protocol for inter-vehicle communication systems. *Proceedings of the 1st ACM International Workshop on Vehicular Ad Hoc Networks (VANET'04)* (pp. 76–85). Philadelphia, PA: ACM.

Krajzewicz et al. (2012). SUMO (Version 0.19.0) [Computer Software]. Retrieved from http://sumo.dlr.de/wiki/Main_Page

Li, G., Wang, W., Yao, X., & Chen, W. (2013). SOBP: a sender-designated opportunistic broadcast protocol for VANET. *Telecommunication Systems, 53*(4), 453-467.

Liu, J., Yang, Z., & Stojmenovic, I. (2013, June). Receiver Consensus: On-Time Warning Delivery for Vehicular Ad-Hoc Networks. *IEEE Transactions on Emerging Topics in Computing, 1*(1), 57-68.

OpenStreetMap contributors. (2015). Planet dump [Data file from $date of database dump$]. Retrieved from http://planet.openstreetmap.org.

Ros, F. J., Ruiz, P. M., & Stojmenovic, I. (2012, January). Acknowledgement-Based Broadcast Protocol for Reliable and Efficient Data Dissemination in Vehicular Ad Hoc Networks. *IEEE Transactions on Mobile Computing, 11*(1), 33-46.

Schwartz, R. S., Barbosa, R. R., Meratnia, N., Heijenk, G., & Scholten, H. (2011, November). A directional data dissemination protocol for vehicular environments. *Computer Communications, 34*(17), 2057–2071.

Schwartz, R. S., Das, K., Scholten, H., & Havinga, P. (2012, June). Exploiting beacons for scalable broadcast data dissemination in VANETs. *Proceedings of the 9th ACM International Workshop on Vehicular Inter-Networking, Systems, and Applications (VANET)* (pp. 53-62). Low Wood Bay, Lake District, UK: ACM.

Schwartz, R. S., Scholten, H., & Havinga, P. (2013, November). A scalable data dissemination protocol for both highway and urban vehicular environments. *EURASIP Journal on Wireless Communications and Networking 2013*, 257. Retrieved from http://jwcn.eurasipjournals.com/content/2013/1/257

Sommer, C., German, R., Dressler, F. (2015) Veins (Version 2.2) [Computer Software]. Retrieved from http://veins.car2x.org/

Stojmenovic, I. (2004, November). Comments and Corrections to "Dominating Sets and Neighbor Elimination-Based Broadcasting Algorithms in Wireless Networks". *IEEE Transactions on Parallel and Distributed Systems, 15*(11), 1054–1055.

Stojmenovic, I., Seddigh, M., & Zunic, J. (2002, January). Dominating sets and neighbor elimination based broadcasting algorithms in wireless networks. *IEEE Transactions on Parallel and Distributed Systems, 13*(1), 14–25.

Suriyapaibonwattana, K., & Pornavalai, C. (2008). An Effective Safety Alert Broadcast Algorithm for VANET. *International Symposium on Communications and Information Technologies*, 247–250.

Suriyapaiboonwattana, K., Pornavalai, C., & Chakraborty, G. (2009). An Adaptive Alert Message Dissemination Protocol for VANET to Improve Road Safety. *FUZZ-IEEE*, 20-24.

Tseng, Y. C., Ni, S. Y., Chen, Y. S., & Sheu, J. P. (1999). The broadcast storm problem in a mobile ad hoc network. *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99)* (pp. 151–162). Seattle, WA: ACM.

Varga, A.(2015). OMNeT++ (Version 4.6) [Computer Software]. Retrieved from https://omnetpp.org/omnetpp

Wisitpongphan, N., Tonguz, O., Parikh, J., Mudalige, P., Bai, F., & Sadekar, V. (2007). Broadcast storm mitigation techniques in vehicular ad hoc networks. *IEEE Wireless Communications*, 84–94.