Palestine Polytechnic University

Deanship of Graduate Studies and Scientific Research

Master of Informatics

A Stable Clustering Algorithm for VANETs

Submitted By

Eng. Omar Baradia

A Thesis Submitted in Partial Fulfillment of Requirements for The Degree of Master's of Informatics

October 24

The undersigned hereby certify that they have read, examined, and recommended to the Deanship of Graduate Studies and Scientific Research at Palestine Polytechnic University the approval of a thesis entitled: **"Hybrid stable clustering algorithm"**, submitted by Omar Ibrahim Baradia in partial fulfillment of the requirements for the degree of Master in Informatics.

Graduate Advisory Committee:

Dr. Liana Tamimi (Supervisor), Palestine Polytechnic University.

Signature:________________ Date:________________

Dr. Radwan Tahboub (Internal committee member), Palestine Polytechnic University.

Signature:________________ Date:________________

Dr. Ahmad S Alsadeh (External committee member).

Signature: [illegible] Date: 22-01-2024

Thesis Approved

Signature:________________ Date:________________

DECLARATION

I declare that the Master’s Thesis entitled "**Hybrid stable clustering algorithm**", is my original work, and hereby certify that unless stated, all work contained within this thesis is my independent research and has not been submitted for the award of any other degree at any institution, except where due acknowledgment is made in the text.

Eng. Omar Baradia

Signature: [illegible] Date: 24/10/2023.

DEDICATION

I dedicate my dissertation work to my family and many friends. A special feeling of gratitude to my loving parents, Ibrahim Baradia and Arwa Abufara whose words of encouragement and push for tenacity ring in my ears. My brothers have never left my side and are very special, especially Khalid Baradiah for helping me develop my programing skills.

I also dedicate this dissertation to my many friends who have supported me throughout the process. I will always appreciate all they have done.

ACKNOWLEDGEMENT

I would like to express my deep gratitude and respect to the people who helped me in completing this work. I would like to express thanks to my supervisor, Dr. Liana Al Tamimi, for her support and guidance throughout the work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

.

LIST OF ABBREVIATIONS

Symbol	**Description**
CAM	Cooperative Awareness Message
CATRB	Clustering algorithm using the traffic regularity of buses
CBSC	Center-Based **Secure** and Stable Clustering Algorithm
CECGP	Center-based Stable Evolving Clustering Algorithm with Grid Partitioning
CH	Cluster Head
CM	Cluster Member
DSRC	Dedicated Short-Range Communication
DSSS	Direct Sequence Spread Spectrum
FHSS	Frequency Hopping Spread Spectrum
GPS	Global Positioning System
HSCA	Hybrid Stable Clustering Algorithm
IP	Internet Protocol
ITS	Intelligent Transportation Systems
LAN	Local Area Networks
LLC	Logical Link Control layer
LT	Life Time
LTE	Long Term Evolution
MABSC-CD	Multi-Agent-Based Stable Clustering and Collision Detection
MAC	Medium Access Control
MAN	Metropolitan Area Networks
MANETs	Mobile Ad Hoc Networks
MLME	MAC Layer Management Entity
MPDU	Protocol Data Units

OBU	On Board Unit
OFDM	Orthogonal Frequency Division Modulation
PHY	Physical Layer
PLCP	Physical Layer Convergence Protocol
PLME	Physical Layer Management Entity
PMD	Physical Medium Dependent
QoS	Quality of Service
SUMO	Simulation of Urban Mobility
TDMA	Time Division Multiple Access
UDP	User Datagram Protocol
V2V	Vehicle to Vehicle
VANET	Vehicular Ad Hoc Network
WAVE	Wireless Access in Vehicular Environment
WLAN	Wireless Local Area Network

Abstract

Despite the clear advantages offered by Vehicular Ad-hoc Networks (VANETs) over Mobile Ad-hoc Networks (MANETs), which stem from their ability to regulate vehicular movement according to traffic laws, including actions like stopping, slowing down, and changing lanes, VANETs still grapple with instability issues. These issues result from various factors such as high-speed vehicle behaviors, frequent vehicle joinings and departures, link failures, and alterations in network topology.

Numerous algorithms have been developed to enhance network stability in terms of maintaining vehicles within a cluster for extended periods, whether they serve as cluster heads or cluster members, minimizing packet loss rates, and achieving this stability with minimal message and packet exchanges to avoid network bottlenecks. Most of these algorithms focus on organizing and clustering vehicles based on their mobility patterns.

Stability means the continuity of communication between vehicles on the one hand and the base station on the other hand, thus delivering packets close to real time, and thus an almost non-existent rate of data loss. In the proposed algorithm, this means a group of vehicles that travel together at close distances and in the same direction along the distance traveled.

This thesis primarily investigates two well-known clustering algorithms: the Center-Based Secure and Stable Clustering algorithm (CBSC) and the Scalable Clustering algorithm (SCalE). CBSC determines clustering centers based on vehicle density to maximize cluster size, while also considering vehicle speed, acceleration, and distance between vehicles for cluster head selection. However, CBSC overlooks vehicle behaviors that could lead to link disruptions when vehicles make turns or exit highways. On the other hand, SCalE aims to address these issues by using vehicle behavior as a criterion for cluster head selection, focusing on stable vehicles that are less likely to leave the road. It employs differences in speeds and positions to choose cluster heads, ultimately extending the cluster members' lifetimes.

However, the authors of SCalE did not account for scenarios where vehicles travel in opposite directions, briefly coming close to each other and the cluster head before moving away, which affects cluster duration. To address this, the proposed Hybrid Stable Clustering Algorithm (HSCA) was introduced. HSCA strives to create clusters where vehicles travel closely together for longer distances and durations. It uses vehicle density to determine cluster centers and adopts the cluster head selection method from SCalE, focusing on vehicle behavior while disregarding vehicles traveling in different directions.

HSCA has demonstrated its effectiveness across various simulation scenarios, including changes in the number of vehicles, their speeds, and cluster ranges. Despite resulting in more cluster compared to the previous algorithms, HSCA maintains consistent result quality due to its focus on selecting cluster elements based on behavior and movement direction. Additionally, changes in vehicle speeds and locations are accommodated within the cluster head election process, ensuring adaptability to different simulation scenarios.

We did not mention the security aspects. Although we meant to talk about Wi-Fi, we mentioned it because communication between vehicles is based on this technology.

Based on the experiments that were conducted, it was found that the proposed algorithm (HSCA) was able to outperform its counterparts (SCalE and CBSC) in the improvement rate. It was able to outperform SCALE in all cases that were tested with a high percentage, and it was also able to outperform the CBSC algorithm in most scenarios, and we clarified Reasons why it is not superior in some scenarios.

The proposed algorithm achieved an improvement rate of 45% in terms of CH life time compared to CBSC when changing the number of nodes, and 196% compared to SCALE.

Likewise, in CM life time, HSCA achieved an improvement of 89% compared to CBSC and 230% improvement compared to SCALE. When implementing the scenario of changing the number of nodes and knowing the impact of this on the time required to form a cluster, the HSCA algorithm achieved a better improvement compared to CBSC and SCALE by 22% and 46.46.85%, respectively. The proposed algorithm was able to outperform SCALE in terms of CH and CM life time under the cluster range change, while the results were close to CBSC, where the improvement rate was 117% and 19%, respectively. The same was true for CM life time, where the percentage was 57% and 2.9%, respectively. While the effect of mobility speed was remarkable, the proposed algorithm achieved an improvement of 187% compared to SCale in terms of CH life time and 116% in terms of CH life time.

Keywords: VANET, CM, CH, SCalE, CBSC, HSCA.

Chapter 1

Introduction

In this chapter, we will review the work methodology in this thesis, starting with defining the problem statement and the motive for submitting it. We started this chapter with an overview of the thesis in the section 1.1, indicating the research problem statement, objectives, benefits, and main contributions are discussed in section 1.2 to 1.5, and finally in section 1.6 we briefly outline the main structure of the thesis.

1.1 Thesis Overview

Vehicular Ad-hoc Networks (VANETs) represent a cutting-edge technological advancement that continues to evolve. These networks have played a pivotal role in enhancing intelligent transportation systems, enabling the integration of modern technologies into vehicles. This includes features like GPS based tracking systems, robust communication networks, and direct communication capabilities among vehicles facilitated by a specialized unit called the On Board Unit (OBU) [4].

These specialized modules enable vehicles equipped with them to communicate in various ways. They use vehicle-to-vehicle (V2V) technology for communication between vehicles, vehicle-to-infrastructure (V2I) technology, and even vehicle-to-roadside communication [1]. Vehicular Ad-hoc Networks (VANETs) are a crucial component of Intelligent Transportation Systems (ITS). [16] They play a vital role in solving many challenges and aid decision-making processes in various scenarios, with traffic congestion and emergency route planning being among the most significant [6].

In VANETs, communication methods can generally be divided into two categories based on the radio interfaces they rely on [1]. One category uses dedicated short-range communication (DSRC) as its foundation, while the other leverages existing cellular technologies. Moreover, thanks to advancements like Long Term Evolution (LTE) and delivery technology, these networks are constantly evolving to ensure scalability, performance efficiency, stability, and security. This is particularly important due to the rapid progress in communication technologies and the integration of LTE networks and handover technology. To enhance network performance, clustering vehicles that use these networks together has been widely recognized as an effective solution [1]. The primary aim of this approach is to manage the flow of data between vehicles. However, because vehicles are highly mobile and dynamic, they tend to reorganize into clusters frequently, which can challenge the stability of these clusters.

There are various types and clustering in VANET. Topology, mobility, and contextual information are all types of information that can be used for VANET clustering. Both V2V and V2I can be used to collect this information either locally or globally. Researchers have been recently encouraged to explore the possibility of leveraging V2I to create efficient, stable, and high-performance VANET clusters through rapid development and expansion of road infrastructure [2].

Clustering technology has been used to improve connectivity by grouping nodes in a geographic area together based on their mobility [2].

This thesis proposes a hybrid clustering technique in which the CBSC and SCalE algorithms are combined in the process of selecting Cluster Head (CH), Cluster Member (CM) distribution, and forming strong clusters. The new algorithm, called HSCA, has been adapted to combine the features of the two algorithms.

Where selection index is used to determine the centers of clusters instead of the Metric (M) used in the CBSC algorithm. In this case, we ensure that we overcome the link rupture problem that occurs when two vehicles meet in opposite directions, which helps increase life time.

In addition to relying on cluster head back up in the SCalE algorithm in order to speed up the maintenance process, a custom mobility helper was developed to know the behavior of each node, and the Global Positioning System (GPS) was achieved through the mobility model [12], as this model contains a parameter to determine the speed of each vehicle and another parameter to determine the vehicle's coordinates.

Wireless Fidelity (WiFi) technology (IEEE 802.11) was used to send CAM (Cooperative Awareness Message) messages between vehicles every one second to identify neighboring vehicles in order to maintain vehicle communication between each other and ensure that each vehicle follows the CH [6].

Several experiments have been performed with change in parameter, the results were taken at different numbers of nodes, different mobility speed for nodes and different range of cluster. The HSCA algorithm is evaluated using the NS-3 simulation system. The results showed using the advantages of CBSC in locating clusters and SCalE suitability metric in the proposed algorithm reduced cluster head changes which approved cluster head lifetime more than CBSC and more than SCalE. In addition to almost non-existent loss of packages due to the long time of the vehicle within the cluster, and the location of the vehicle played a major role in determining the cluster to which the vehicle belongs, so an increase in the number of nodes leads to an increase of cluster head life time and Cluster member life time , this is due to the increase in the size of the cluster and the density of the distribution of nodes, also the accurate factors used to elect suitable cluster head , so packets loss will be minimal and cluster formation will be as fast as possible but will take less than CBSC and SCalE algorithms.

1.2 Problem Statement and Its Significance

Despite the restrictions imposed by Vehicular Ad-hoc Networks (VANETs) to regulate the movement of vehicles, these networks were not able to reach the required degree of stability, as there is still data loss, loss of communication between network elements and other problems caused by the behavior of vehicles in this network, which increase with Network expansion, and development of communication technologies provided by these networks the issue of arriving at a clustering model that achieves the desired goals of using security networks for reliability, security and stability has become an important issue for research [4].

When we talk about stability, one of the most important factors of network stability depends on how long vehicles stay within a single cluster. The longer they stay, the more stable the network is, and the longer the cluster lives, the more stable the network becomes [1]. Current algorithms show different performance when we talk about how long a vehicle stays. Most stable as a cluster leader (CH life time), and the same is true for CM lifetime.

The CBSC algorithm showed a noticeable effect on improving network stability, but it failed to outperform the SCalE algorithm in terms of CH lifetime and CM lifetime. Therefore, these two algorithms can be made more efficient by combining their characteristics into one algorithm in order to reach the best results and achieve stability.

1.3 Research Objectives

The main purpose of our research is to build a stable clustering algorithm that outperforms CBSC and SCalE and is more stable as follows:
This work aims to:

- Increase CH lifetime rate.
- Increase CM lifetime rate.
- Increase stability by reducing data loss and increasing time per cluster.
- Reduce the packet loss rate.

1.4 Research Methodology

Our Methodology can be summarized as:

- Studying current clustering algorithms and identifying their advantages and disadvantages.
- Proposing a hybrid algorithm to improve VANET network performance.
- Choosing a suitable simulator and simulating the algorithm.
- Assessing the proposed algorithm performance by contrasting it with the CBSC and SCalE algorithms.

1.5 Research Contributions

The main contributions of this thesis:

- Proposing hybrid approach combining CBSC and SCalE algorithms and adopt the behavior of vehicles and the direction of movement in clustering process.
- Increase stability, as stability means the continuity of communication between vehicles on the one hand and the base station on the other hand [3], thus delivering packets close to real time, and thus an almost non-existent rate of data loss.
- Choosing cluster centers based on the density of vehicles, thus covering the largest possible number of nodes (vehicles).
- Choosing a cluster head based on precise criteria directly related to the nature of the mobility of VANET network vehicles, which is a major reason for the instability of this type of network.
- Reducing the time for choosing a cluster head by appointing a backup cluster head.
- Reducing the number of changes and interruptions in the network by relying on the behavior and direction of vehicle movement in choosing the cluster head.
- Building a simulation model to test the new algorithm and compare it with other algorithms.

1.6 Thesis Organization

In chapter 2, we did some background research and looked at what others have studied. We also talked more about wireless technology, and the fundamentals of VANETs, compared VANETs and MANETs again, and summarized the main characteristics of these networks. In chapter 3, we took a deep dive into three different methods: CBSC, SCalE, and our own hybrid algorithm. We tested them in different situations with various settings to see how they performed. We also explained how these methods work. In chapter 4 we created a virtual environment to simulate how our new algorithm performs. We collected data and compared it to how CBSC and SCalE do. This helped us figure out if our method was better. In chapter 5, we summarize everything we have learned in this thesis. We drew conclusions based on our research and suggested what could be studied in the future**.**

Chapter 2

Background and Literature Review

In this chapter, present two types of Ad-hoc Networks. Mobile Ad-hoc Networks (MANET) and Vehicular Ad-hoc Networks (VANETs), the advantages, characteristics, applications and types of communications, for each network and the difference between them.

The primary objective of this study is to examine the essential approaches and strategies utilized to improve the reliability and functionality of the VANET network, particularly in difficult conditions. Furthermore, it aims to investigate the specific functions of various network components and evaluate their influence on the overall stability and performance of the network. By conducting this analysis, a comprehensive comprehension of the network's behavior and the factors that affect its stability and performance can be acquired.

This literature review concentrates on analyzing multiple algorithms to identify primary areas for improvement, aiming to enhance stability in the VANET network. Additionally, it explores the potential of implementing clustering methods to boost stability, control, and effectiveness in information exchange.

By examining these aspects, the study aims to provide valuable insights into strategies that can optimize the stability and overall performance of the VANET network.

2.1 Wireless Technology

Over the last few years, the technology for wireless communications has made tremendous advantages. It allows very high mobility, efficient working and is almost extremely economical.

Wireless technologies are categorized into two main groups: large-area technologies and local area technologies. Large-area technologies, such as Global System for Mobile communication (GSM), General Packet Radio Service (GPRS), or Universal Mobile Telecommunication Service (UMTS), offer moderate bandwidth. On the other hand, local area technologies like WLAN (Wireless Local Area Network) provide a much higher bandwidth. This thesis will primarily focus on WLAN, the latter category. There are two distinct standards for Wireless LAN: HIPERLAN, established by the European

Telecommunications Standards Institute (ETSI), and 802.11, developed by the Institute of Electrical and Electronics Engineers (IEEE). Currently, the 802.11 standard overwhelmingly dominates the market, with well-engineered implementing hardware. Therefore, it is advisable to concentrate on this particular standard [7].

• The IEEE 802 family

The WLAN protocols of the IEEE 802.11, as referenced in [7], are included in the 802 family, which establishes standardization for both Local Area Networks (LAN) and metropolitan area networks (MAN). The 802 family incorporates a shared Logical Link Control layer (LLC), which is standardized in 802.2. The network layer, typically utilizing the Internet Protocol (IP) and its routing protocols such as Ad-hoc On-demand Distance Vector (AODV) or Dynamic Source Routing protocol (DSR) for mobile Ad-hoc-networks (as illustrated in Figure 2.1), is positioned on top of the LLC [8].

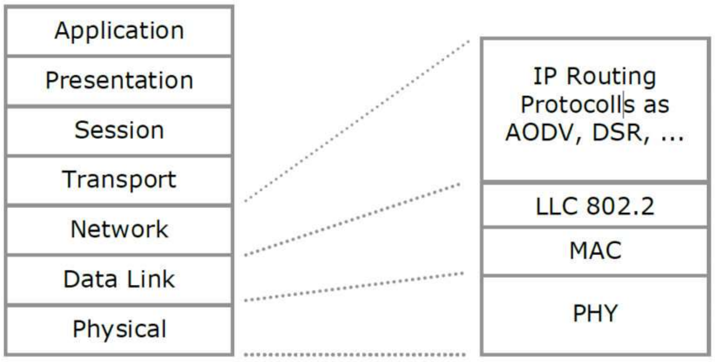


Figure 2.1: ISO/OSI layer model [8]

The MAC and PHY layers, which are part of the Media Access Control and physical layer respectively, are combined within the same standard subgroup beneath the LLC. Numerous standard subgroups exist for Ethernet and wireless LAN, as specified in 802.11, as illustrated in Figure 2.2.

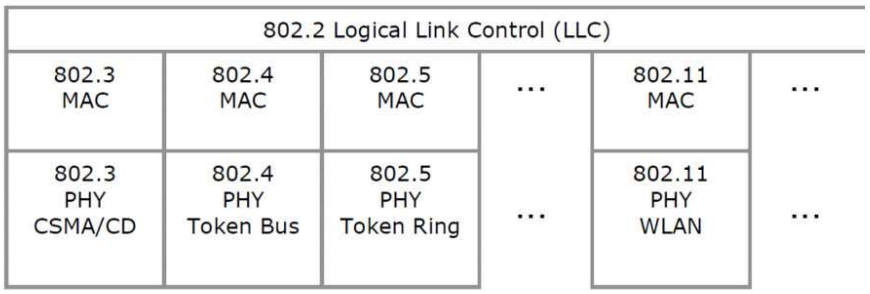


Figure 2.2: 802 LLC, MAC and PHY [8]

• The Mac Layer

The MAC layer consists of a series of protocols that are in charge of overseeing and organizing the utilization of a shared medium. The Station Management Entity (SME) and the MAC Layer Management Entity (MLME) are responsible for controlling the MAC layer [7].

• The Operation Modes

The IEEE 802.11 standards, specifies two different ways to configure a network: Ad-hoc and infrastructure.

The infrastructure mode employs stationary network access points, as depicted in Figure 2.3, to facilitate communication among mobile nodes. Typically, these access points are linked to landlines to enhance the local area network's capacity by connecting wireless nodes to other wired nodes. In the event of overlapping service areas of access points, mobile nodes may be transferred between them. This configuration bears a striking resemblance to contemporary cellular networks worldwide.

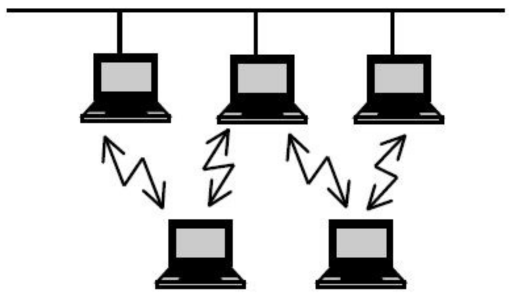


Figure 2.3: WLAN infrastructure mode

In an Ad-hoc network, computers are assembled to establish a network spontaneously. As depicted in Figure 2.4, the network lacks a predetermined structure, fixed points, and typically allows each node to communicate with any other node within its communication range. These networks are commonly referred to as Mobile Ad-hoc Networks (MANET).

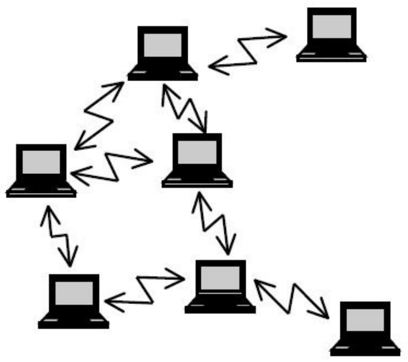


Figure 2.4: WLAN Ad-hoc mode

• The PHY Layer

The physical layer itself can again be divided into two parts:

- The Physical Layer Convergence Protocol (PLCP).
- The Physical Medium Dependent (PMD).

Responsible for the control of these sublayers is the Physical Layer Management Entity (PLME).

The Physical Layer Convergence Protocol (PLCP) provides a mechanism for converting the MAC sublayer protocol data Units (MPDU) into a framing format that is suitable for the transmission and reception of data and management information using the associated Physical Medium Dependent (PMD) system. Additionally, the PLCP is responsible for functions such as carrier sensing, clear channel assessment, and basic error correction.

The PMD directly interacts with the physical medium and performs fundamental bit transmission functions within the network. Its primary responsibilities include encoding and modulation. Spread spectrum technologies are employed to mitigate the susceptibility of the signal to narrowband interference and frequency-dependent fading. These technologies expand the narrowband signal into a broadband signal using a specialized code. In older systems, Frequency Hopping Spread Spectrum (FHSS) has been utilized, while newer systems implement Direct Sequence Spread Spectrum (DSSS) or Orthogonal Frequency Division Modulation (OFDM) [8].

2.2 Fundamentals of Vehicular Ad-hoc Networks (VANETs)

In this section, we will explore the fundamental concepts and terminology related to VANETs, prior to delving into the presentation of basic clustering algorithms and techniques. The concepts discussed include Ad-hoc networks, the distinctions between VANETs and MANETs, as well as their potential applications. The subsequent discussion highlights the necessity for clustering and its relevance to VANETs. Lastly, a concise overview of the fundamental techniques for retrieving a vehicle's position is provided.

One could even contemplate the possibility of combining these two modes into a hybrid network structure. This would enable the provision of internet access to a large number of mobile nodes through a limited number of base stations. However, it is important to note that there is currently no established standard for such a hybrid mode.

Introduction to Ad-hoc Networks

The term "Ad-hoc network" pertains to wireless networks that are decentralized, as illustrated in Figure 2.5. The fundamental characteristic of such networks is that they do not rely on a preexisting infrastructure. An Ad-hoc network is a collection of devices that possess equal status on the network and are free to associate with any other network device within the link range. During the communication process between nodes, the participants of the same network typically need to establish a backbone of the network before attempting to exchange messages with each other. Clustering is a well-known technique for creating a backbone of the network .

Alternatively, communication could be made efficient through an excessive exchange of broadcast messages, known as 'flooding.' However, this technique would inundate the network with redundant messages and lead to contention and collision problems. Clustering aims to create groups of nodes in such a way that they can communicate with each other with the minimum possible exchange of redundant messages. Before proceeding, it is necessary to analyze what decentralization means and what advantages this technique offers in Ad-hoc networks [10].

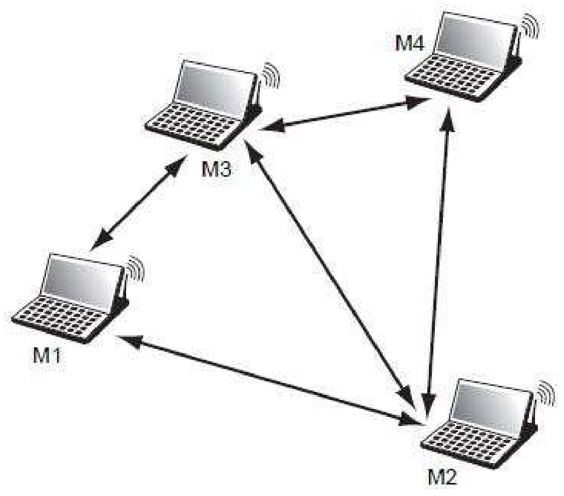


Figure 2.5: An example of Ad-hoc network [6]

Networks can be categorized into two primary classifications based on their organizational structure. They can either be centralized or decentralized.

A centralized network is one where the planning and decision-making activities, such as clustering or other issues, are concentrated within a specific node or group of nodes. In this type of network, these nodes possess a high level of awareness regarding the exact topology of the rest of the network. On the other hand, decentralized networks are characterized by the dispersion of decision-making to nearly all nodes within the network. It is evident that the centralized version is unsuitable for Ad-hoc networks, particularly due to the absence of preexisting infrastructure. Therefore, it is impossible for a specific node to have knowledge of the exact position of all other nodes, which becomes even more challenging when considering node mobility [7].

The decentralized nature of Ad-hoc networks makes them well-suited for various applications where reliance on central nodes is not feasible. Firstly, the scalability of wireless Ad-hoc networks compared to wireless managed networks must be considered. The minimal configuration and quick deployment of Ad-hoc networks make them suitable for emergency situations such as natural disasters or military conflicts, as well as in locations where network infrastructure cannot be taken for granted. The presence of dynamic and adaptive routing protocols enables the rapid formation of Ad-hoc networks. However, it is important to note that there are theoretical and practical limitations to the overall capacity of such networks [7][8].

MANETs and VANETs

MANET is an abbreviation for mobile Ad-hoc network. A MANET is a network formed by a group of mobile devices that are connected wirelessly, without relying on a fixed infrastructure. These devices have the ability to self-configure and can move freely in any direction, resulting in frequent changes in the network links. Additionally, each device in the network acts as a router, forwarding traffic that is not related to its own use. Therefore, it is necessary for each device to have a mechanism for maintaining and updating this information in order to effectively route traffic. MANETs can operate independently or be connected to the larger Internet. It is important to note that MANETs may experience frequent network disconnections, particularly in areas with low vehicle density. The high mobility of the nodes in such networks can also result in strict delays in message transmission [18].

Numerous algorithms have been proposed for Mobile Ad-hoc Networks (MANETs) with the aim of enhancing the stability of the routing infrastructure and reducing the number of exchanged messages between nodes, as well as minimizing packet drops resulting from erroneous routing decisions, among other factors [1].

Vehicular Ad-hoc Networks (VANETs) are a specific type of Mobile Ad-hoc Networks (MANETs). The term VANET refers to a network of vehicles that communicate with each other in an Ad-hoc manner, similar to MANETs, which are characterized by their rapid and dynamic changes. Both VANETs and MANETs lack a pre-established infrastructure and central control. Furthermore, devices in both networks are free to move in any direction, but in VANETs, movement is restricted to predefined roads. Additionally, the speed of vehicles in VANETs is limited by factors such as speed limits, congestion levels, and traffic control mechanisms, such as traffic lights and stop signs [18].

It is worth noting that future vehicles may be equipped with advanced capabilities, such as antennas with longer transmission ranges, extensive on-board storage capacities, more processing power, and rechargeable sources of energy. These capabilities can be easily achieved in VANETs but not in MANETs. Table 2.1 below presents a comparative analysis of the main characteristics of VANETs and MANETs.

Table 2.1: Summary of the basic characteristic of MANETs and VANETs [18].

Parameter	MANET	VANET
Cost of production	Cheap	Expensive
Change of the network topology	Slow	Frequent and very fast
Mobility	Low	High
Node density	Sparse	Dense and frequently variable
Bandwidth	Hundred kps	Thousand kps
Range	Up to 100m	Up to 500m
Node lifetime	Depends on power resource	Depends on lifetime of vehicle
Moving pattern on nodes	Random	Regular
Position acquisition	Using ultrasonic	Using GPS, RADAR

Possible applications on VANETs

There are several appealing applications for VANETs. Firstly, it is worth mentioning that due to the similarities between VANETs and MANETs, many applications of MANETs can also be utilized for VANETs. However, there are differences in the specific details. In VANETs, vehicles must proceed in an organized manner and are restricted to roadsides, as opposed to the random movement of vehicles in MANETs. The interactions with roadside equipment can also be accurately characterized [1] [18].

One of the most crucial applications is the enhancement of driving safety by assisting drivers in avoiding collisions and providing alternative routes between highway entries and intersections. This can be easily implemented using GPS and navigation systems. Additionally, drivers can receive traffic jam reports, high-speed tolling information, and the fastest routes to their destinations. Future applications could include cruise control systems that automatically adjust to maintain safe distances between vehicles or alert drivers of emergency vehicles in the vicinity [4].

Another highly useful application is the ability to transform a traffic jam into productive work time by allowing drivers to download and read emails. The system could also enable Voice over Internet Protocol (VoIP) services such as Google-Talk or Skype for communication between employees.

Furthermore, Intravehicular Communications (IVC) can be utilized to provide comfort applications such as file exchange, interactive communications, weather information, and the locations of gas stations or restaurants near the requester. In addition to IVC, Road Vehicle Communications (RVC) applications can also be implemented.
In conclusion, the potential applications of VANETs encompass a wide range of areas including traffic management, road monitoring, entertainment, and contextual information [6].

Techniques for Retrieving Position

Prior to proceeding, it is imperative to acknowledge that all algorithms necessitate access to certain position information in order to operate effectively. Each vehicle must possess either a proficient method for ascertaining its absolute position within the confines of a city or road framework, or its relative position in relation to other vehicles. The most renowned techniques employed for this purpose include:

- GPS.
- Doppler effect from where we export the **Doppler Value.**
- Received Signal Strength (RSS).

On the following algorithms, the technique that was used to retrieve position information is GPS.

Clustering

As mentioned earlier, VANET are networks without a fixed infrastructure, and the precise layout of the network is not known to every node within it. Therefore, in order to prevent excessive message flooding, it is necessary to establish a backbone for the network prior to vehicles initiating message exchanges. In our particular scenario, we aim to create an abstract framework over the network, allowing regional modifications to remain imperceptible to the entire network. This objective can be accomplished through the implementation of clusters.

Clustering is essentially the procedure of forming specific subgroups within the overall network structure. To provide a more precise definition, a paper titled "A Survey of Clustering Schemes for Mobile Ad-hoc Networks" [17] describes it as follows: "In a clustering scheme, mobile nodes are organized into distinct virtual groups, and they are grouped together based on their geographic proximity within the same cluster. Specific rules determine how nodes within a cluster behave differently from those outside the cluster." This process of grouping nodes based on certain criteria helps organize and manage the network effectively.

All the nodes are classified to one of the following 3 categories which is also shown in the Figure 2.6:

1. **Cluster heads** of the cluster: they have to rebroadcast the message to every member of the group and they act as the local coordinators for their cluster,
2. **Cluster members**: they just receive the message within the cluster without rebroadcasting it to anyone else.
3. **Orphan nodes (standalone):** which are mainly ex-members of a cluster that currently do not belong to any cluster.

In some classification algorithms we may find gateways: they rebroadcast the message to their neighboring heads of the clusters and they are used for communication between clusters.

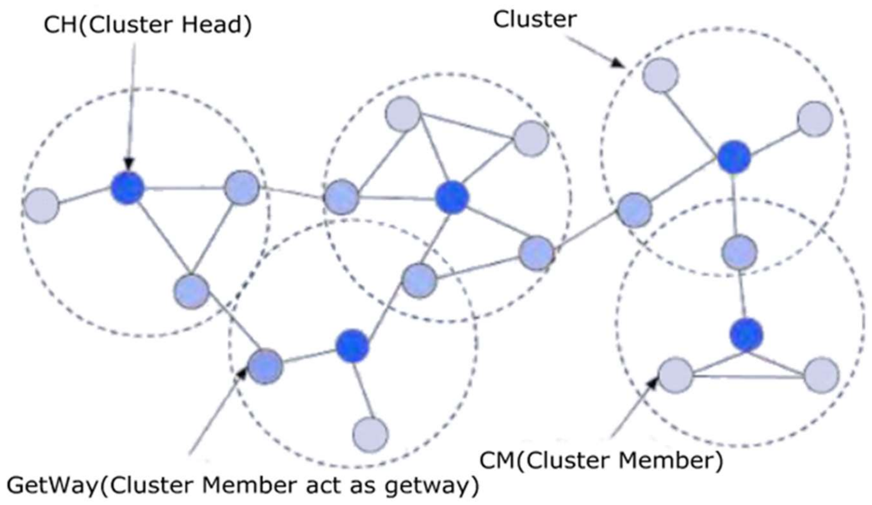


Figure 2.6: Clustering, the fundamentals.

The fundamental benefits of clustering can be succinctly outlined as follows:

1. The spatial reutilization of resources, wherein two clusters that are not adjacent to each other can employ the same frequency or code on a non-overlapping multicast structure [5].

2. The utilization of a virtual backbone, comprised of cluster heads and gateways, to prevent excessive data flooding [5].

3. The enhanced stability and scalability of the network, particularly for regular members, as local modifications do not necessitate visibility and updating across the entire network [6].

In addition, it is important to consider the cost associated with creating and maintaining clusters in such networks when discussing the advantages of clustering. Firstly, it should be noted that in order to maintain clustering structures, nodes in certain pairs must exchange messages periodically. The frequency of these message exchanges varies depending on the mobility patterns of the nodes. However, in a rapidly changing network, a sufficient number of messages must be exchanged, which leads to energy and bandwidth consumption [6].

Secondly, certain cluster schemes may require the complete rebuilding of the cluster structure throughout the entire network when certain local events occur, such as a node leaving (referred to as the ripple effect or re-clustering) [1][6].

Thirdly, many clustering schemes operate in two phases: cluster formation and cluster maintenance. Some of these schemes assume that during the cluster formation phase, the nodes maintain their positions, which is not accurate in realistic scenarios. In this case, the time required for cluster formation, known as the computation round, must also be taken into consideration. This time can vary depending on the algorithm, and the nodes cannot remain stationary for an extended period [1].

There are numerous algorithms proposed for clustering in Mobile Ad-hoc Networks (MANETs) and Vehicular Ad-hoc Networks (VANETs). It is important to mention that protocols designed for MANETs are not suitable for VANETs [18]. This is because MANET protocols do not consider the unique characteristics of VANETs, particularly in terms of movement patterns, making them unsuitable for intravehicular communication. MANET routing protocols can be classified into three major categories:

1. Proactive protocols: These protocols maintain and update routing information between all nodes in a network at all times.

2. Reactive protocols: Route determination is invoked on a demand or need basis.

3. Hybrid protocols: These protocols combine proactive and reactive elements. An example of such a protocol is the zone routing protocol.

In our case, the reactive algorithms are more suitable in order to reduce flooding messages.

A beacon based algorithm is the only applicable kind of algorithm on VANETs. In VANET, stability pertains to the network's ability to maintain its connectivity and performance despite node mobility, network congestion, and other factors that may affect its operation. Clustering plays a crucial role in enhancing the stability of the collaborative environment by reducing communication overhead and improving the network's scalability and reliability. Clustering algorithms can group vehicles based on location, speed, and other parameters, thereby reducing the number of messages transmitted and improving network performance [5].

Cluster stability is often affected by cluster fragmentation and size [3]. Given the importance of network stability and its ability to handle traffic promptly, it is prioritized as a measure for evaluating cluster performance. To achieve optimal performance, the cluster must exhibit stability, less overhead, long CH, and long CMs [4].

This study focuses on the methods and techniques employed to develop a stable and usable VANET network, even under adverse conditions. It examines the role of each network component and its impact on overall stability and performance. The paper investigates several algorithms to identify areas for improvement and increase stability in the VANET network. It also explores how clustering can contribute to stability, control, and effective information exchange.

The performance metric for stability in VANET CH life time, CM life time, average cluster head change, packet loss, and cluster formation time are important performance metrics in VANET that affect the stability and efficiency of communication [1].

The following performance metrics are crucial in assessing the efficacy of clustering algorithms and protocols in VANET:

- Cluster Head Lifetime: This metric represents the duration during which a node fulfills the role of a cluster head. A prolonged cluster head lifetime results in a more stable network, as frequent alterations in cluster heads have the potential to disrupt communication and escalate overhead [6].

- Cluster Member Lifetime: This metric signifies the duration for which a node maintains its membership within a cluster. A prolonged lifespan as a cluster member also enhances network stability, as frequent alterations in cluster members can lead to communication disruptions and heightened overhead [6].

- Average Cluster Head Change: This metric denotes the frequency at which a node transitions between its roles as a cluster head and a cluster member, or vice versa. A substantial average cluster head change indicates network instability, as frequent alterations in cluster heads can impede communication and escalate overhead [6].

- Packet Loss: This metric represents the proportion of packets that are lost during the process of transmission. A substantial packet loss rate signifies network instability, as the loss of packets can lead to delays, retransmissions, and heightened overhead [6].

- Cluster Formation Time: This metric denotes the time taken for a cluster to form. A longer cluster formation time can indicate network instability, as it can lead to communication delays and increased overhead [6].

By optimizing these performance metrics, it is possible to enhance the stability and efficiency of communication in VANET and ensure reliable and timely message delivery [1][3].

The stability means the continuity of communication between vehicles on the one hand and the base station on the other hand, thus delivering packets close to real time, and thus an almost non-existent rate of data loss. In the proposed algorithm, this means a group of vehicles that travel together at close distances and in the same direction along the distance traveled [15].

2.3 Literature review

Cluster algorithms in VANET can be classified into two primary categories: cluster establishment with CH selection and cluster formation, and cluster maintenance. The former type focuses on the initial creation of clusters and the selection of appropriate CHs. The latter type, cluster maintenance, involves ensuring the ongoing stability and efficiency of the clusters once they have been formed [2].

By implementing these clustering algorithms, the VANET network can enhance communication, improve coordination, and optimize information exchange among nodes within clusters, thereby leading to enhanced overall network performance and stability [15]. The stability of the cluster is enhanced in cluster maintenance through the prediction of link failures among the nodes.

A CH is selected by determining the network parameters, while the other nodes are considered as cluster members. The CH assumes full responsibility for internal cluster communication. The internal communication within a cluster utilizes two specific routing methods known as intra cluster communication and internal cluster communication. Additionally, the formation of clusters also reduces the excessive load on the nodes in the VANET. The cluster plays a crucial role in radio transmission by accessing channels that minimize collisions and interference in communication information [1].

• Stable clustering algorithms

In this section, a succinct overview of the algorithms studied and their operational mechanisms will be presented.

Communication in Vehicular Ad-hoc Networks (VANETs) can be broadly classified into two groups based on the radio interfaces employed. One group of approaches relies on Dedicated Short-Range Communication (DSRC) as the underlying technology. The other group involves the utilization of existing cellular technologies that are already in use. Given the rapid advancements in mobile cellular networks, some researchers propose leveraging the existing infrastructure and technologies for VANET communications. Various techniques have been suggested to effectively manage VANETs and make use of the established mobile cellular networks for data transmission. As a result, there has been a quest for a dependable clustering approach in VANETs to mitigate frequent cluster reformation [1] [15].

A. ***CBSC algorithm****:*

In cluster-based Vehicular Ad-hoc Networks (VANETs), a clustering algorithm is utilized to establish clusters from a group of unlabeled nodes. Within this framework, all vehicles present in the clusters transmit their data to a base station referred to as an evolved Node B (eNodeB), which is an integral part of the cellular network infrastructure. The management of the clusters is undertaken by the eNodeB. To facilitate effective communication between the eNodeB and the CMs, a CH serves as an intermediary. Within a cluster, a vehicle assumes the role of a CH and gathers data from all CMs using the 802.11p communication standard. Subsequently, the CH establishes communication with the eNodeB utilizing a technology such as TLE [1].

1. Cluster head selection

A relative mobility metric, denoted as M, has been devised for the purpose of selecting an optimal Cluster Head (CH) in CH elections. The objective is to identify a CH that can effectively prolong the lifespan of the cluster and minimize the frequency of CH reselection. The relative mobility metric evaluates disparities in terms of maximum acceleration, maximum speed, and relative location between a given vehicle and all other vehicles within the same cluster. A smaller value of M, speed, and acceleration indicates a lower relative mobility compared to other vehicles in the cluster. These factors are incorporated into the CH selection metric to enhance cluster stability and reduce the occurrence of CH reselection [1].

2. Cluster Maintenance and Reforming

The transient nature of cluster existence can be ascribed to the capricious and dynamic nature of traffic. The recurrent or instantaneous reconstitution of clusters is not a viable remedy. The authors in [1] proffer a cluster maintenance algorithm to alleviate the frequency and expense of cluster reformation.

B. ***CEC-GP algorithm****:*

The research conducted by the Center-based Stable Evolving Clustering Algorithm with Grid Partitioning (CEC-GP) [2] has placed a greater emphasis on clustering in VANETs. Some existing methods for clustering VANETs utilize statistical clustering techniques such as K-means or other models that incorporate density measurements.

This study specifically focuses on research that utilizes base stations on the road, which are increasingly prevalent in today's infrastructure, to perform clustering in VANETs. The aim is to leverage the expanding range of V2I communication for LTE and to enable a global view-based clustering approach, which enhances stability and predictability in decision-making. To achieve this, the paper explores the clustering of VANETs in a highway environment using a center-based approach.

The initial stage of data summarization, known as the grid-partitioning step, plays a crucial role in dividing the environment into grids. This step serves as the fundamental basis for the approach, which incorporates vehicle motion information. Each grid serves as a higher-level entity that assists in determining the creation of outliers or clusters during the grouping of information. The approach is outlined as a comprehensive framework for clustering Vehicular Ad-hoc Networks (VANETs), encompassing tasks such as assigning, selecting a cluster head, merging, and eliminating clusters.

C. CATRB algorithm:

The Clustering Algorithm using Traffic Regularity of Buses (CATRB) is a method that considers the mobility characteristics of vehicles, such as velocity, position, and direction, to address the issue at hand [3]. In order to estimate the distribution of other vehicles in space and time, the algorithm also takes into account fixed bus routes. CATRB selects the most suitable Cluster Head (CH) in the Vehicular Ad-hoc Network (VANET) by utilizing the in center, circumcenter, and centroid of an equilateral triangle. By being centrally located within the cluster, the resulting CH effectively manages the entire system. Furthermore, the proximity between the CH and Cluster Members (CMs) allows for convenient and successful data transfer. The authors of this study conducted simulations using real traffic data from Taipei, which demonstrated that the proposed strategy significantly enhances cluster stability by reducing the likelihood of CH changes [3].

In CATRB, a three-stage method is employed to address various issues in VANET. In the first stage, the algorithm records the Traffic Congestion History Table (TCHT) of each Bus Node (BN), as well as the Cluster Head Table (CHT) and Cluster Member Table (CMT) of the vehicle node. In the second stage, a CH election procedure is suggested to minimize cluster fragmentation and enhance stability. BNs determine the ideal CH to achieve this goal. Finally, in the third stage, a secondary CH election process is proposed to reduce cluster instability caused by clusters with an excessive number of nodes [3].

D. ***SC-GDRV algorithm****:*

In the paper titled "A Stable Clustering Approach based on Gaussian Distribution and Relative Velocity in VANETs" (SC-GDRV) [4], a novel approach was proposed for the analysis of clusters in VANETs. This approach utilized the average speed of vehicles and the standard deviation to classify clusters into two distinct levels of stability.

The first level consisted of the most stable clusters, which encompassed approximately 68% of the vehicles. The second level included the less stable clusters, encompassing about 95% of the vehicles. Within these clusters, any vehicle with a velocity equal to or close to the average value could be chosen as the Cluster Head (CH) [4].

By employing this approach, SC-GDRV aimed to establish stable clusters based on a statistical analysis of vehicle velocities. The selection of CHs based on relative velocities and adherence to the average value contributed to enhancing the stability and effectiveness of communication within the VANET network [4].

The objective of this research is to enhance network stability and dynamically reduce the topology. The Gaussian normal distribution principle is utilized to accept and reject compounds during the cluster formation process, in addition to the method of velocities and standard deviation. Clustering is employed to reduce Routing Overhead and improve message delivery. The role of the algorithms is divided into two parts, and each section has its own functions [4]:

- Cluster formation: Create clusters, CH selection, Set CM, Getaway, and Assist Connection.
- Cluster maintenance: Improve the links, describe leaving, joining vehicles, and merging clusters.

The utilization of a normal distribution (also known as a Gaussian distribution) was employed due to its efficacy in representing random phenomena. A significant association exists between the size of the sample and the likelihood of the samples conforming to a Gaussian distribution. By reducing the recommended velocity limit, the stability of the clusters will be enhanced.

The proposed solution is based on creating two types of clusters [4]:

- **Type 1**: The quantity of automobiles present in a designated section of the roadway is computed, and the mean and standard deviation of their velocities are determined. The preponderance of the curve denotes 68% of the vehicles, which are the most consistent, while approximately 32% of the vehicles, which have an impact on the stability of the network, are excluded. The chosen vehicles exhibit similar speeds.
- **Type 2**: This category comprises 95% of the vehicles as per the curve and can be utilized to establish a stable cluster while eliminating 5%. The inter-vehicle velocity in this category is higher than that of the first type.

E. MABSC-CD algorithm:

The Multi-Agent-Based Stable Clustering and Collision Detection (MABSC-CD) technology introduces a novel approach known as the Dynamic Clustering Technique. This technique has been developed to address the issue that arises when a vehicle leaves the Cluster within a network. In addition to resolving traffic congestion problems, the technology also identifies and reports congested areas. Furthermore, a routing protocol algorithm has been devised in the Base Station (BS) to account for mobility and overcome routing problems. In the event of congestion or collision detection, an alternative route is promptly sought, taking into consideration the mobility of the BS node and Sensor Node when making routing decisions [5].

This study has introduced a potential solution for addressing traffic congestion and collisions within the VANET network. A novel technology has been developed to detect instances of traffic collisions, and the weight factor has been utilized to select CH. The efficacy of this new approach has been demonstrated through its ability to minimize data loss and enhance data transfer rate.

F. SCalE algorithm:

The Stable Clustering algorithm for vehicular Ad-hoc networks (SCalE) is introduced, which presents a stable cluster head selection scheme that utilizes the knowledge of the vehicle's behaviors [6]. This algorithm employs a clustering technique where neighboring vehicles are grouped into clusters, with one vehicle elected as a Cluster Head (CH) in each cluster. Within these clusters, each vehicle (Cluster Member CM) communicates with Unmanned Aerial Vehicles (UAVs) or base stations through the CH of their respective cluster. The algorithm incorporates two key features: the utilization of the vehicle's behavior knowledge to efficiently select CHs, and the inclusion of a backup CH to ensure the stability of the cluster structures.

• Summary and discussion

In this section, a comparative analysis is presented in Table 2.2, outlining the merits and demerits of each algorithm under study. Furthermore, Table 2.3 delves into the algorithms studied and compared, with a focus on the three most crucial factors that impact stability in VANETs networks, namely, CH duration, CM duration, and the number of clusters in each network.

As the network is more stable when the value of CH duration and CM duration increases and the value of Number of clusters decreases.

Table 2.2: Summary of previous studies (Advantages & Disadvantages)

Algorithm	Advantages	Disadvantages
CBSC [1]	1-Enhance the dependability and efficacy of communication. 2-Furnishes a cluster maintenance algorithm to curtail the frequency and expense of cluster reforming, thereby augmenting the steadiness of the system.	It has been observed that it does not surpass ScalE in terms of the average CH lifetime.
CEC-GP [2]	1-The inclusion of multi-level data aggregation enables improved clustering decisions through the reduction and summarization of data. 2-The integration of base stations along the road facilitates efficient, reliable, and high-performing clustering in VANETs.	The study fails to consider the expenses associated with the implementation of the suggested clustering model, which could be substantial and impede its practical utility.
CATRB [3]	1-Enhances cluster stability through a reduction in the frequency of cluster head transitions. 2-Incorporates the mobility attributes of vehicles, including velocity, position, and direction, to augment cluster stability.	1-The algorithm requires each vehicle to calculate the route with the minimum cost for every destination in the network. 2-This calculation can be time-consuming and may cause a delay in the transmission of data.
SC-GDRV [4]	1-Reducing congestion and enhancing message delivery. 2-Increase the network stability and reduce the topology dynamic by integrating the normal probability distribution function.	1-The proposed scheme was evaluated using a vehicle system consisting of only 21 vehicles, which may limit the generalizability of the results.
MABSC-CD [5]	1-Improved head life of the cluster 2-Increased stability, especially in difficult cases such as congestion and traffic accidents. 3-Addresses the problem of traffic congestion and detects and reports congested areas.	1-Power consumption is high in the proposed method.
SCalE [6]	1-Enhance the cluster stability in average CH lifetime and reduce the data loss rate.	1-It needs a complex infrastructure in order to be able to get all the requirements of this algorithm. 2-Compared to CBSC the average Life Time of CM was lower.

Table 2.3: Summary of previous studies (Metrics Performance)

Algorithm	CH Duration	CM Duration	# of Clusters
CBSC	Medium	High	Low
CEC-GP	Medium	High	Low
CATRB	Low	Medium	Medium
SC-GDRV	High	Low	Low
MABSC-CD	Low	Low	Low
ScalE	High	Low	Medium

Upon comparison between CBSC and CEC-GP, it is apparent that these two algorithms exhibit similar performance characteristics. However, comprehensive testing has revealed that the performance of CEC-GP deteriorates as traffic density increases. In contrast, CBSC utilizes image processing techniques, making it more responsive to dense clusters. Consequently, as traffic density intensifies, the CH (Cluster Head) duration proves to be more advantageous in CBSC. Additionally, as per the findings from experiments conducted by the research team at [1], CBSC demonstrated complete superiority over SCalE. However, it fell short of surpassing SCalE in terms of CH lifetime.

In the CATRB algorithm, a new algorithm was not developed to reduce the number of clusters. Instead, a new algorithm was employed to expedite the selection of a new CH each time, and thus, the issue of CH duration remained unresolved.

Regarding the SC-GDRV algorithm, there has been an enhancement in the duration of CH (Cluster Head) due to the highly efficient method employed in selecting the most stable compound to serve as CH. However, it is important to note that this method primarily relies on clustering vehicles with similar speeds together, which can result in a substantial increase in the number of clusters or a decline in the quality of the resulting cluster. Furthermore, the effectiveness of this method is dependent on the speed of data processing, with a higher number of vehicles and their speeds leading to increased energy consumption during cluster formation, thereby causing instability.

On the other hand, the MABSC-CD algorithm has succeeded in reducing the number of clusters by implementing a dynamic cluster that can adapt its radius and size based on the speed and density of the vehicles. However, it has not shown improvement in the lifetime of CH and CM (Cluster Member), as the author has indicated a significant decrease in CH duration with increasing speed.

According to the findings presented in Table 2.3, it is evident that there is no single category that can be universally applied to all network scenarios. For instance, while the

CBSC algorithm effectively enhances stability in VANETs networks, it encounters challenges in terms of infrastructure requirements, as it necessitates advanced technology in each vehicle. Additionally, the control of the Cluster Head time rate poses a potential issue, which can have a detrimental impact on network stability in certain cases. Consequently, it is advisable to combine the strengths of multiple algorithms to create a hybrid and dynamic approach that can adapt to the overall network situation.

This survey aims to investigate stable clustering algorithms with the objective of identifying the most suitable algorithm that maintains crucial metrics for network stability. The algorithms are categorized based on their ability to sustain the lifetime of Cluster Heads (CH) and Cluster Members (CM). Algorithms that exhibit higher capabilities in preserving these factors generally have longer average cluster lifetimes. Additional metrics considered in the survey include data loss rate and average cluster formation time for each algorithm.

The survey thoroughly examines the advantages and disadvantages of each algorithm in terms of their requirements, performance, and methodology. Table 2.3 clearly demonstrates that the CBSC and SCalE algorithms outperform other algorithms across most parameters. However, it is worth noting that CBSC falls short in terms of CH lifetime when compared to SCalE, resulting in a higher average CH change and reduced stability for CBSC.

Summary of the chapter

The technology used in VANET networks was discussed, the most important basics of the network were studied, and the difference between VANET and MANET networks was learned. Then the clustering process, its importance and use were discussed. Then a review of the literature was conducted on a set of algorithms for the clustering process, while examining the pros and cons of each one. These algorithms, and suggestions for future improvement methods.

Chapter 3

Methodology and Implemented Algorithms

3.1 Overview

A clustering algorithm must endeavor to uphold cluster stability and preserve the cluster contents and structure for an extended period. Failure to do so will result in frequent re-clustering processes, which will ultimately diminish the communication's performance.

In the upcoming sections, we will delve into the details of three distinct algorithms. The first algorithm, known as the Center-Based Secure and Stable Clustering algorithm (CBSC), is introduced in [9] CBSC employs direction vectors, identifies the centers of denser vehicle areas, and utilizes intersections to form a smaller number of more stable clusters.

The second algorithm, the Stable Clustering Algorithm for vehicular Ad-hoc networks (SCalE), is presented in the paper [10]. SCalE incorporates stable nodes into the leader election process by assessing node behavior (whether they will remain or leave) and maintaining a backup of qualified cluster header candidates.

The third algorithm is the proposed algorithm, called Hybrid Stable Clustering Algorithm (HSCA), which acts as (CBSC) at the two phases. The first phase is cluster formation (initial clustering) and the second one is cluster maintenance (progressive). It will be implemented only the algorithm related to retrieval of a node's position through GPS, but the difference at cluster head election, where will follow SCalE head election method.

This chapter is devoted to the exposition of the three algorithms that were previously mentioned. Firstly, the algorithm with fundamental procedures, equations, and figures is presented. Subsequently, notes pertaining to its implementation are presented alongside the algorithm.

3.2 A Center-Based Secure and Stable Clustering Algorithm

The proposed algorithm, known as CBSC (Center-Based Secure and Stable Clustering), is a distributed clustering algorithm based on mobility. Its main objective is to prioritize cluster stability. By stability, we refer to the coherence and consistency within clusters [1]:

- Long cluster head duration with low rate of cluster head change.
- Long cluster member duration.

The aforementioned objective shall be attained through the establishment of clusters, wherein the cluster heads and their respective members shall maintain the minimum distance and minimum relative velocity. The algorithm employed for this purpose utilizes GPS technology to furnish the positional data of the vehicles.

Security is not the scope of the thesis; we did not mention the security aspects. Although we meant to talk about Wi-Fi, we mentioned it because communication between vehicles is based on this technology.

CBSC

A clustering algorithm forms clusters out of a collection of unlabeled nodes. All cars in cluster-based VANETs deliver their data to a base station used in cellular network infrastructure evolved Node B (eNodeB). The vehicles are then managed by clusters using eNodeB. To facilitate communication between eNodeB and CMs, a CH serves as a messenger. A vehicle serves as a CH in a cluster, gathering data from all CMs through 802.11p and communicating with the eNodeB over TLE. In the study [1], the authors made the following assumptions regarding cluster:

- Both 802.11p and LTE interfaces are incorporated within each vehicle.
- GPS devices have been installed in each vehicle, thereby providing precise geolocation information.
- Every vehicle is aware of its destination, maximum speed, and acceleration.

In [1] provides a center detection-based clustering approach based on these assumptions.

- **Cluster Head Selection**

A relative mobility metric M is developed for CH election in order to choose an optimal CH that can lengthen the cluster lifetime and lower the frequency of CH reselection. The relative mobility metric measures the differences in maximal acceleration, maximum speed, and relative location between one vehicle and every other vehicle in the same cluster. A lower relative mobility than other vehicles in this cluster is indicated by a smaller M, speed and acceleration, are added to the CH selection metric to make the cluster stabled and decrease the CH reselection frequency [1]. To select an appropriate CH:

- **Step1:** For a vehicle k, which is in the cluster cluster_i, the position difference between it and all other N vehicles in the same cluster cluster_i is:

$$D_k = \sum_{n=1}^{N} \sqrt{(x_k - x_n)^2 + x(y_k - y_n)^2} \, ..(1) \quad [1]$$

- **Step 2:** The speed difference between K and all other N vehicles in the same cluster is:

$$V_k = \sum_{n=1}^{N} |v_k - v_m| \,..(2) \ [1]$$

- **Step 3:** maximal acceleration difference between k and all other N vehicles in the same cluster is:

$$A_k = \sum_{n=1}^{N} |a_k - a_m| \,..(3) \ [1]$$

The relative mobility metric M is:

$$M_k = \alpha \frac{D_k}{max\{D_n|\forall n \in C_i\}} + \beta \frac{V_k}{max\{V_n|\forall n \in C_i\}} + \gamma \frac{A_k}{max\{A_n|\forall n \in C_i\}} \,..(4) \ [1]$$

Where α, β, and γ are weighted coefficients. $\alpha+\beta+\gamma = 1$.

They can be adjusted to fit the different traffic conditions. When the traffic condition is good and all vehicles are driving at a similar speed, the distance between vehicles has a greater effect. Thus, the value of α should be higher than the other two. When vehicles are driving at high speed, the value of β is higher than the other two. When the vehicles enter an area which speed limit changes continually, a higher γ should be considered.

The relative mobility metric Mevaluates the relative position, speed, and maximal acceleration differences between one vehicle and all other vehicles in the same cluster. A smaller M indicates the vehicle has lower relative mobility than other vehicles in this cluster.

Algorithm 1: Cluster head selection algorithm in CBSC [1]

```
Input: Vehicles in one cluster
Output: Cluster head o of the corresponding cluster
Set M $min$ = +∞;
for each $vehicle\ k$ do:
    Calculate the relative mobility metric M_$k$;
    If $M_k < M_{min}$ then
        M$min$ = M$k$;
        K=0;
    end
end
return 0;
```

So, the CBSC algorithm behaves as in algorithm 2.

Algorithm 2: Clustering Algorithm in CBSC [1]

```
Input: Vehicle set V
Output : initial clusters
Initialize center set C = Ø;
Locate the centers and add them into C;
Initialize point set P= C;
while P # Ø do
    for each point p in P do
        Initialize node set cluster = Ø ;
        for each vehicle in V do
            if $d_{ep}$ <R then
                add e into $cluster_p$
                Remove e from V;
            end
        end
        if $cluster_p \neq Ø$ then
                Call cluster head selection algorithm;
                Return set $cluster_p$;
        end
        Add the intersection nearest to $p$ which meets the conditions into $P$;
    end
    Remove $p$ from $P$;
end
while v ≠Ø do
    for each point c in C do
        Select an element e in V nearest to c;
        Initialize set $cluster_e= \{e\}$;
        Remove e from V;
        Set e as CH;
        for each vehicle v in V do
            if $d_{ev} \leq R$ then
                Add V into $cluster_e$;
                Remove V from $V$;
            end
        end
        Return $cluster_e$;
    end
end
```

- **Cluster Maintenance and Reforming**

The cluster lifetime is merely brief due to the unpredictable and mobile nature of traffic. Reforming clusters regularly or in real time is not practical. It suggests a cluster maintenance algorithm to reduce cluster reforming's frequency and cost. The cluster maintenance process is divided into four steps:

1. No Connections between CH and CM.

2. No Connections between eNodeB and CH.
3. A Vehicle Joins the Network.
4. Two Clusters Are Too Close.

Algorithm3: Cluster maintenance algorithm in CBSC [1]

```
Input: Initial clusters and vehicle set V
Output: New clusters
if the eNodeB cannot reach a CH then
    Call Cluster Head Selection Algorithm;
end
if the CH cannot reach a CM then
    Remove the CM;
    Notice eNodeB;
end
if the distance between two CHs $\leq R$ for a period $\Delta t$ then
    Merge the two clusters into one cluster;
    Call the Cluster Head Selected Algorithm;
end
if a CM cannot reach the CH then
    if it can receive a signal from CHs then
        Join the cluster whose signal of CH is strongest;
    end
    else
        Notice eNodeB;
        The node performs as a CH;
    end
end
```

- **Notes about the algorithm:**

In order to increase the stability of this algorithm, the direction of movement of the vehicle must be taken into consideration. So, to demonstrate the advantage of grouping vehicles, an example presented on Figure 3.1.

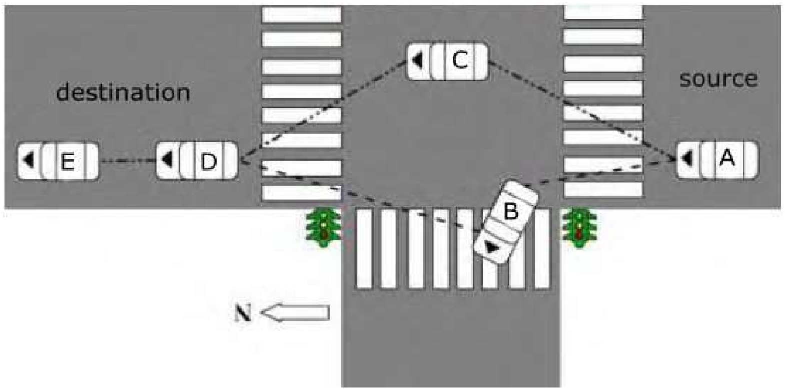


Figure 3.1: Link rupture example [1]

Link rupture event is more likely to occur between vehicles A, B and D. Analytically, five vehicles are moving towards an intersection and vehicle B is turning onto a new street, whereas the other 4 vehicles are continuing straight on the same road. A connection is established between vehicles A and E. Communication is possible on two routes: one via vehicle B (route A-B-D-E) and the other via vehicle C (route A-C-D-E). As vehicle B is turning left and vehicle A is continuing straight, the former route is more likely to be ruptured after a certain time. Consequently, the selection of the second route is more suitable and adds more stability and reliability to the communication path between the 2 vehicles (A and F).

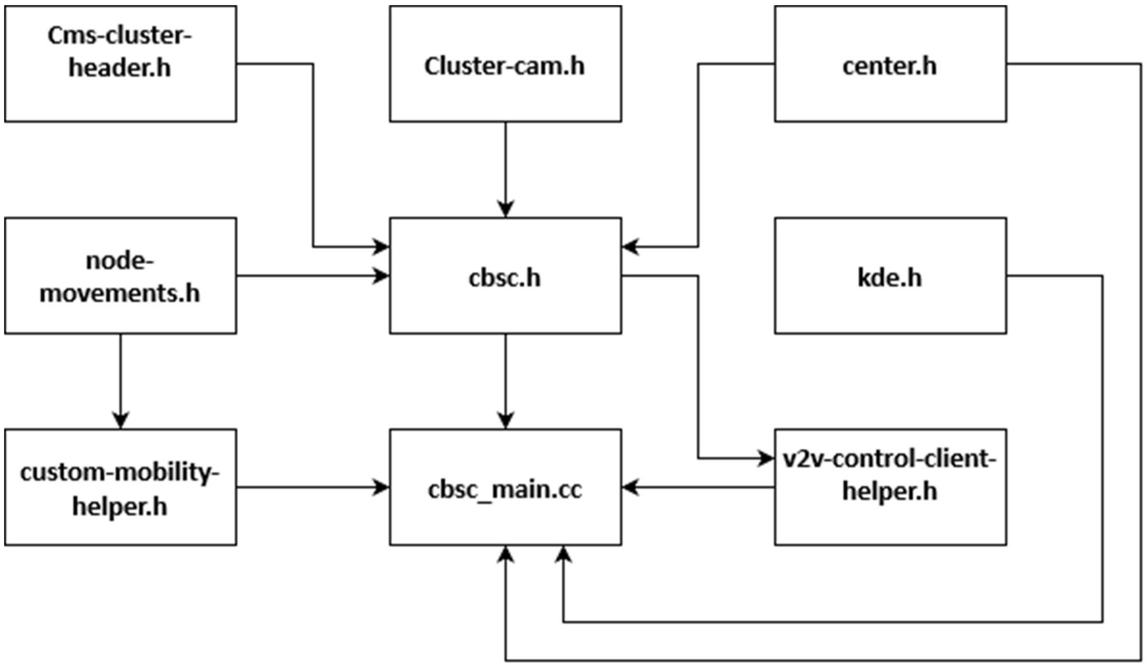


Figure 3.2: Files that were implemented in order to simulate the CBSC algorithm and how they were included.

3.3 Stable Clustering Algorithm for vehicular Ad-hoc networks (SCalE)

This clustering algorithm groups neighboring vehicles into a cluster and selects two of them as the cluster head and backup cluster head, respectively. The stability of the cluster structures is achieved by the use of knowledge of the vehicle’s behavior in the cluster-head selection as well as the use of the backup cluster head to enable efficient maintenance of the cluster structure.

Presenting a stable cluster head selection scheme that is achieved using the knowledge of the vehicle’s behaviors [6].

This Algorithm uses a clustering technique; neighboring vehicles are grouped into one cluster, with a particular vehicle elected as a Cluster Head (CH) in each cluster. Each vehicle (Cluster Member CM) in this cluster communicates with Unmanned Aerial Vehicles (UAVs) or base stations through the CH of the associated cluster. Two features are incorporated into the algorithm:

1. Knowledge of the vehicle's behavior for efficient selection of CHs.
2. Employment of a backup CH to maintain the stability of cluster structures.

- **Cluster Head election:**

The choice of CH is based on index Sigh K (ξk). It is an indicator that is calculated by exchanging messages between the network and each K vehicle, this message is known as Cooperative Awareness Message (CAM) and containing information relevant to safety related applications (vehicle speed and position) and the vehicle with the lowest index ξk is chosen as CH and the remaining compounds are CMs.

- **Step 1:** Vehicle Behavior (B_k) the value is set to 1 if the vehicle intends to leave the system and to 0 otherwise.

$$B_k = \begin{cases} 1, & if\ k\ is\ leaving\ the\ road\ way \\ 0, & if\ k\ is\ not\ leaving\ the\ road\ way \end{cases} ..(1)\ [6]$$

It is used to filter out unstable vehicles as cluster head.

- **Step 2:** Relative Mean Speed (S_k): The stability of the clusters can degrade rapidly in a highly mobile environment. It represents a measure of the stability of a vehicle in a VANET. It calculates by take the difference in Velocities (V) between the vehicle (k) and all Neighboring (N) vehicles within its range, the value is between 0 and 1.

$$S_k = \frac{\sum_{n=1}^{N} |v_k - v_n|}{N \cdot max\{\Omega k\}} ..(2)\ [6]$$

Where the normalizing factor is the maximum value of the set Ωk. This is defined as the set of all the vehicles speed differences $|v_k - v_n|$within the set Φ_k, provided the vehicles are moving (v > 0), and is formally expressed as Ωk = {$|v_k - v_n|$ | (v_k, v_n) > 0; ∀n E Φ_k}$..(3)$ [6].

- **Step 3:** Mean relative distance (D_k)): indicates that the neighboring vehicles are closer to the potential CH. by GPS coordinates of two vehicles k and n ,we can calculate the distance between the two at an arbitrary time as [6] :

$$\Delta x_{k,n} = |x_k - x_n| ..(4)$$
$$\Delta y_{k,n} = |y_k - y_n| ..(5)$$

Consequently, the mean relative distance D_k of vehicle k is defined as the mean Euclidean distance. Furthermore, normalizing by the maximum value of the set Z_k makes S_k and D_k comparable:

$$D_k = \frac{\sum_{n=1}^{N} \sqrt{\left[\Delta x_{k,n}\right]^2 + \left[\Delta y_{k,n}\right]^2}}{N \cdot max\{Z_k\}} ..(6)\ [6]$$

The set Z_k is composed of all the Euclidean distances between the vehicles belonging to the set Φ_k, that is:

$$Z_k = \left\{\sqrt{\left[\Delta x_{k,n}\right]^2 + \left[\Delta y_{k,n}\right]^2} | \forall n \in \Phi_k | \right\} ..(7)\ [6]$$

- **Step 4:** The value of ξ_k is calculated by adding $S_k + D_k$, and when the CAM is periodically exchanged between vehicles, each vehicle records the values and exchanges them with other vehicles, and the vehicle with the lowest index becomes CH.

$$\xi_k = S_k + D_k ..(8)$$

Aa such will always fall in the range [0 , 2]. Upon periodical exchange of CAMs amongst all the vehicles in the system, the k^{th} vehicle can record a list of all CH selection indexes ξ belonging to every nth vehicle in its neighbor's set Φ_k. The set of all ξ for every neighbor's set Φ_k is therefore defined as:

$$\Psi_k = \{\xi_k | \forall n \in \Phi_k\} ..(9)$$

Denoting (k) as the ID of the vehicle k, the vehicle will be selected CH if its CH selection index ξ_k is found to be smaller than ξ_n, the selection index of any other vehicle n in range, that belongs to the set Φ_k:

$$CH = \{\gamma_k | \xi(\gamma_n) \leq min\{\Psi_k\}\} ..(10)$$

So, Cluster Head Election Algorithm:

Algorithm 4: Cluster head election algorithm in SCalE [6]

```
Require: ¥k in the system that do not belong to a cluster yet
for each kth vehicle do
    Evaluate $\xi_k$ using (2), (6) and (8)
end for
for each kth vehicle do
    if k is in range of a CHi then
        k $\longrightarrow$ CMi
    else
    Start CH election process:
        Eliminate unstable vehicles with (1)
        if (10) then
          k $\longrightarrow$ CHi
        end if
    end if
end for
go to CLUSTER MAINTENANCE
```

- **Cluster Maintenance or Reforming:**

The rest of the maintenance phase is described in Cluster Maintenance Algorithm, which is designed to minimize cluster changes for every possible event, namely for the following situations:

1. The CH leaves the network, that is the vehicle will turn at the next intersection.
2. The CH is within the communication range of at least another CH.
3. A CM loses connection with the CH.
4. A new vehicle joins the network.

If a CH leaves the system, it will put its CH_{Bkp} in charge of the cluster. All the CMs in cluster Θ_i can therefore still hold onto the original cluster structure and avoid going through the clustering process again. In the case a CH_i can hear at least another CH_i but the backup cluster head of cluster Θ_i, CH^i_{Bkp}, is not in range of the other CH_i, then the CH^i_{Bkp} will become the new CH_i, without the need of a new election.

Algorithm5: Cluster Maintenance Algorithm in SCalE [6]

```
for each $CH_i$ do
    $CH_i$ chooses the $CH^i_{Bkp}$
end for
if $CH_i$ leaves system then
    $CH^i_{Bkp} \longrightarrow CH_i$
end if
if $CH_i$ is in range of another $CH_i$ then
    if $CH^i_{Bkp}$ is not in range of $CH_j$ then
        $CH^i_{Bkp} \longrightarrow CH_i$
    else
        merge cluster $\Theta_i$ and $\Theta_j$
    end if
end if
if $CM_i$ is not in range of $CH_i$then
    go to CLUSTER HEAD ELECTION
end if
if new vehicle k enters the system then
    go to CLUSTER HEAD ELECTION
end if
```

- **Notes about the algorithm*:***

What if vehicle search for a new potential strong cluster head, and finds it, but flow at opposite direction? This algorithm addresses the problem of nodes leaving the road, but does not address the problem of the previous scenario.

For the above scenario, if the vehicle is joined to the cluster, it will not be long-lived in this cluster. Therefore, the direction of the vehicle must be taken into account.

- **Files produced and description**

1. **<algorithm name>_main.cc**: As described in the section (5.5) Initial files and description. There is a call of the SCalE algorithm start application function at the end of the file.
2. **node-movements.h and node-movements.cc:** As described in the section (5.5) Initial files and description
3. **custom-mobility-helper.h and custom-mobility-helper.cc:** As described in the section (**5.5**) Initial files and description.
4. **v2v-control-client-helper.h and v2v-control-client-helper.cc:** As described in the section (5.5) Initial files and description, and pass simulation parameters to SCalE algorithm application.
5. **Cluster-cam.h and cluster.cc:** These two files implement the class "ClusterCam" which represent nodes degree (CH, CM, standalone) and node nighbours information(imsi, clusterId, speed, position,address, chaddress).
6. **Cms-cluster-header.h & Cms-cluster-header.cc:** These two files implement the classes of packet meta data header as:
    1. CMSClusterInfoHeader: Packet meta data header to exchange info with other nodes.
    2. CMSIntiateClusterHeader: Packet meta data header to initialize cluster.
    3. CMSFormClusterHeader: Packet meta data header to start head election algorithm.
    4. CMSCheckConnection: Packet meta data header to check connection with other nodes and update nighbours.
    5. CMSMaintenance: Packet meta data header to reform /maintenance cluster.
7. **scale.h and scale.cc:** These two files implement the class "SCalEAlgorithm"

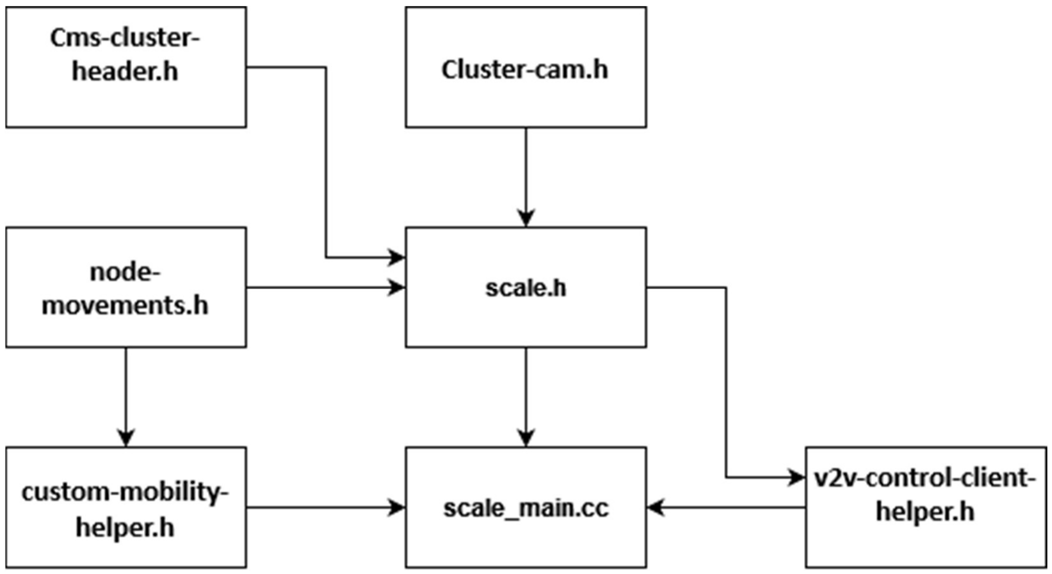


Figure 3.3: Files that were implemented in order to simulate the SCalE Algorithm and how they were included.

3.4 Hybrid Stable Clustering Algorithm for Vehicular Ad-hoc Networks (HSCA)

This algorithm has been proposed in this work to take the advantages of the two previous algorithms and improve the problems they face, so this clustering algorithm groups neighboring vehicles into a cluster and selects two of them as the cluster head and cluster head backup cluster head, respectively.

The stability of the cluster structures is achieved by selecting only neighbors with the same direction and the use of knowledge of the vehicle's behavior in the cluster-head selection as well as the use of the backup cluster head to enable efficient maintenance of the cluster structure.

The proposed algorithm will work according to the CBSC algorithm in all cases and an improved method will be used for the Election of CH as the current method depends on the periodic re-election procedure, the proposed algorithm will work on Adaptive re-election Parameters for CH election used in SCalE will be used instead of those used in CBSC, because based on the study of both algorithms, it has been found that the method used to select the CH in the SCalE algorithm was based on more accurate factors and the method of data processing and calculation is better.

- **Locate centers of clusters:**

After receiving the beacon messages, the system analyzes vehicles' position information and detects the centers of the ranges where the vehicle density is higher than in other areas and without overlap with other centers.

- **Initialize vehicle cluster**:

  Every vehicle in a range of the cluster center then it will be a member of this cluster.

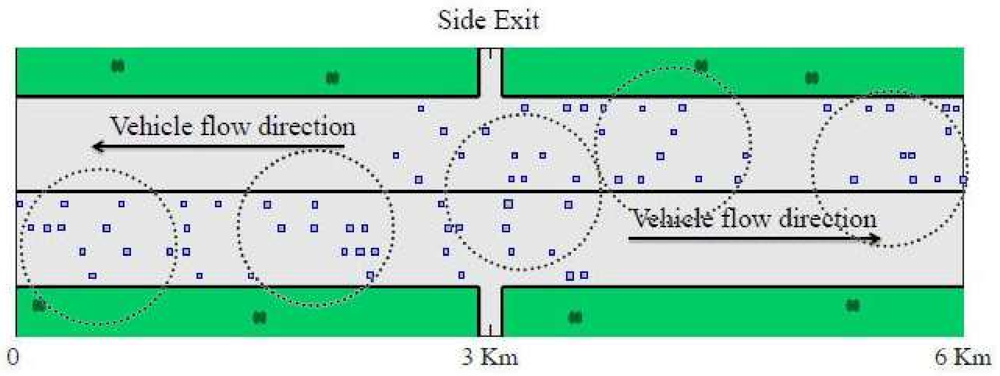


Figure 3.4: Initialize vehicle cluster [6]

- **Cluster Head election:**
- **Step 1:** Vehicle Direction (V_{kd}): The direction type is decided by the angle from the current position to the destination. For vehicle k, whose destination position is (x'_k, y'_k), the direction angle θ_k is

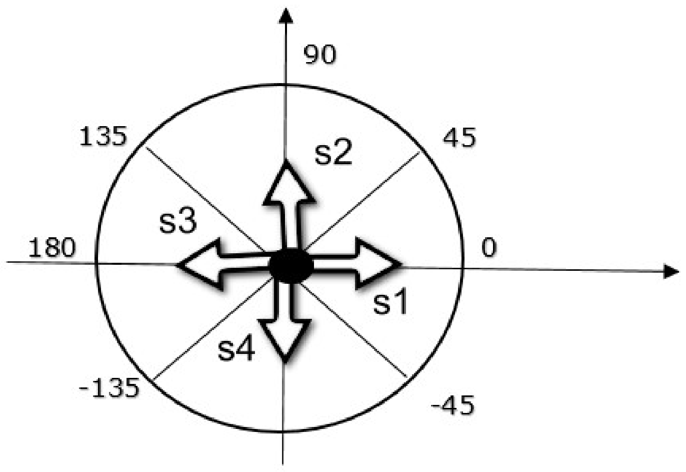


Figure 3.5: The direction angle θ_k between two positions

$$V_{kd} = \begin{cases} 1, \phi_k \in [-45, 45[\\ 2, \phi_k \in [45, 135[\\ 3, \phi_k \in [135, -135[\\ 4, \phi_k \in [-45, -135[\end{cases} \quad ..(1)$$

- **Step 2:** Vehicle Behavior (B_k) the value is set to 1 if the vehicle intends to leave the system and to 0 otherwise. For vehicle k, whose initial position (x_k, y_k) and destination. Position (last position in generated trace file) is $(x'_k,\ y'_k)$, the direction angle θ_k is

$$B_k = \begin{cases} 1\,, \emptyset_k \in [45, 135[\ |\ \emptyset_k \in [-45, -135[\\ 0\,, otherwise \end{cases} ..(2)\ [6]$$

- **Step 3:** Relative Mean Speed (S_k): The stability of the clusters can degrade rapidly in a highly mobile environment. It represents a measure of the stability of a vehicle in a VANET. It calculates by taking the difference in velocities between the vehicle k and all Neighboring N vehicles within its range, the value is between 0 and 1.

$$S_k = \frac{\sum_{n=1}^{N} |v_k - v_n|}{N \cdot max\{\Omega_k\}} ..(3)[6]$$

Where the normalizing factor is the maximum value of the set Ω_k. This is defined as the set of all the vehicle's speed differences $|v_k - v_n|$ within the set Φ_k, provided the vehicles are moving (v > 0), and is formally expressed as:

$$\Omega_k = \{|v_k - v_n|\ |\ (\mathrm{v}_k, \mathrm{v}_n) > 0;\ \forall n \exists \Phi_k\} ..(4)$$

- **Step 3:** Mean relative distance (D_k): indicates that the neighboring vehicles are closer to the potential CH. By GPS coordinates of two vehicles k and n, we can calculate the distance between the two at an arbitrary time as :

$$\Delta x_{k,n} = |x_k - x_n| ..(5)$$

$$\Delta y_{k,n} = |y_k - y_n| ..(6)$$

Consequently, the mean relative distance D_k of vehicle k is defined as the mean Euclidean distance. Furthermore, normalizing by the maximum value of the set Z_k makes S_k and D_k comparable:

$$\mathrm{D}_k = \frac{\sum_{n=1}^{N} \sqrt{[\Delta x_{k,n}]^2 + [\Delta y_{k,n}]^2}}{N \cdot max\{\mathrm{Z}_k\}} ..(7)[6]$$

The set Z_k is composed of all the Euclidean distances between the vehicles belonging to the set Φ_k, that is:

$$\mathrm{Z}_k = \left\{ \sqrt{[\Delta x_{k,n}]^2 + [\Delta y_{k,n}]^2}\ |\ \forall n \Phi_k \right\} ..(8)[6]$$

- **Step 4:** The value of ξ_k is calculated by adding $\mathrm{S}_k + D_k$, and when the CAM is periodically exchanged between vehicles, each vehicle records the values and exchanges them with other vehicles, and the vehicle with the lowest index becomes CH.

$$\xi_k = S_k + D_k..(9)[6]$$

In addition, as such will always fall in the range [0, 2]. Upon periodical exchange of CAMs amongst all the vehicles in the system, the kth vehicle can record a list of all CH selection indexes ξ_k belonging to every nth vehicle in its neighbor's set Φ_k. The set of all ξ for every neighbor's set Φ_k is defined as [6]:

$$\Psi_k = \{\xi_k \mid \forall n \Phi_k\}..(10)$$

Denoting k as the ID of the vehicle k, the vehicle will be selected CH if its CH selection index ξ_k is found to be smaller than ξ_n, the selection index of any other vehicle n in range, that belongs to the set Φ_k [6]:

$$CH = \{\gamma_k \mid \xi(\gamma_k) \leq min\{\Psi_k\}\}..(11)$$

So, Cluster Head Election Algorithm:

Algorithm 6: Cluster head election algorithm in HSCA

```
Input: Vehicles in one cluster
Output: Cluster head o of the corresponding cluster
Set $\xi$ $min$ = +∞;
for each vehicle $k$ do
    Calculate the selection index $\xi_k$;
    if $\xi_k < \xi_{min}$ then
        $\xi$ $min$ = $\xi_k$;
        k = 0;
    end
end
return 0;
```

In this algorithm, the [1] and [6] algorithms were combined together in HSCA.

Algorithm 7: Clustering algorithm in HSCA [1]

```
Input: Vehicle set V
Output: initial clusters
Initialize center set C = Ø;
Locate the centers and add them into C;
Initialize point set P= C;
while P # Ø do
    for each point p in P do
    Initialize node set cluster = Ø ;
    for each vehicle in V do
        if $d_{ep}$ < R then
            add e into $cluster_p$
            Remove e from V;
        end
    end
    if $cluster_p \neq Ø$ then
        Call cluster head selection algorithm;
        Return set $cluster_p$;
    end
    Add the intersection nearest to $p$ which meets the conditions into $P$;
    end
    Remove $p$ from $P$;
end
while v ≠ Ø do
    for each point c in C do
        Select an element e in V nearest to c;
        Initialize set $cluster_e = \{e\}$;
        Remove e from V;
        Set e as CH;
        For each vehicle v in $V$ do
            if $d_{ev} \leq R$ then
                Add V into $cluster_e$
                Remove V from $V$;
            end
        end
        Return $cluster_e$;
    end
end
```

- **Cluster Maintenance and Reforming**

The cluster lifetime is merely brief due to the unpredictable and mobile nature of traffic. Reforming clusters regularly or in real time is not practical. In [1] suggest a cluster maintenance algorithm to reduce cluster reforming's frequency and cost. The cluster maintenance process is divided into four steps:

1. No Connections between CH and CM

2. No Connections between eNodeB and CH
3. A Vehicle Joins the Network.
4. Two Clusters Are Too Close.

Algorithm 8: Cluster maintenance algorithm in HSCA [1]

```
Input: Initial clusters and vehicle set V
Output: New clusters
if the eNodeB cannot reach a CH then
    Call Cluster Head Selection Algorithm;
end
if the CH cannot reach a CM then
    Reomve the CM;
    Notice eNodeB;
end
if the distance between two CHs ≤ $R$ for a period $\Delta t$ then
    Merge the two clusters into one cluster;
    Call the Cluster Head Selected Algorithm;
end
if a CM cannot reach the CH then
    if it can receive a signal from CHs then
        Join the clusterwhose signal of CH is strongest;
    end
    else
        Notice eNodeB;
        The node performs as a CH;
    end
end
```

- **Notes about the algorithm:**

This algorithm addressed the problem of vehicles leaving the road and improved the accuracy of the metric used for cluster head election. However, more stability can be obtained by clustering vehicles traveling in the same direction by choosing cluster centers, for each direction separately or reducing the cluster range so that it is at most equal to the width of the four lanes in each direction as shown in Figure 3.6.

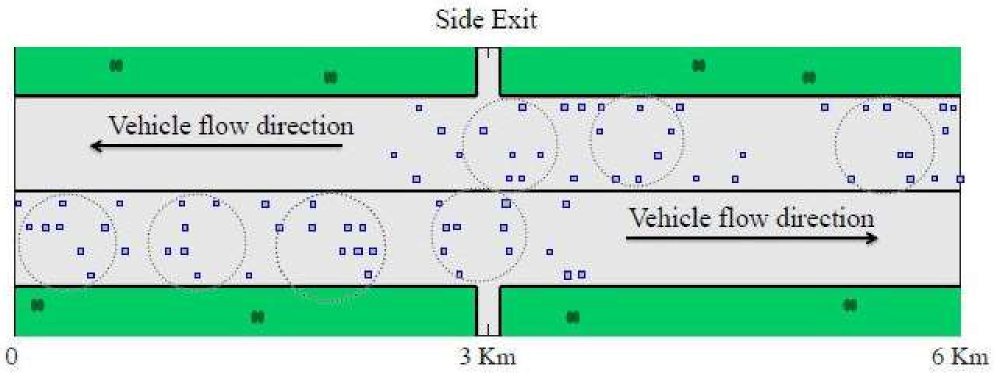


Figure 3.6: Proposed solution for more stability in the HSCA algorithm

So stable vehicles will travel together with cluster members (CH & CM) Lifetimes will become longer, and the number of vehicle state changes will be less.

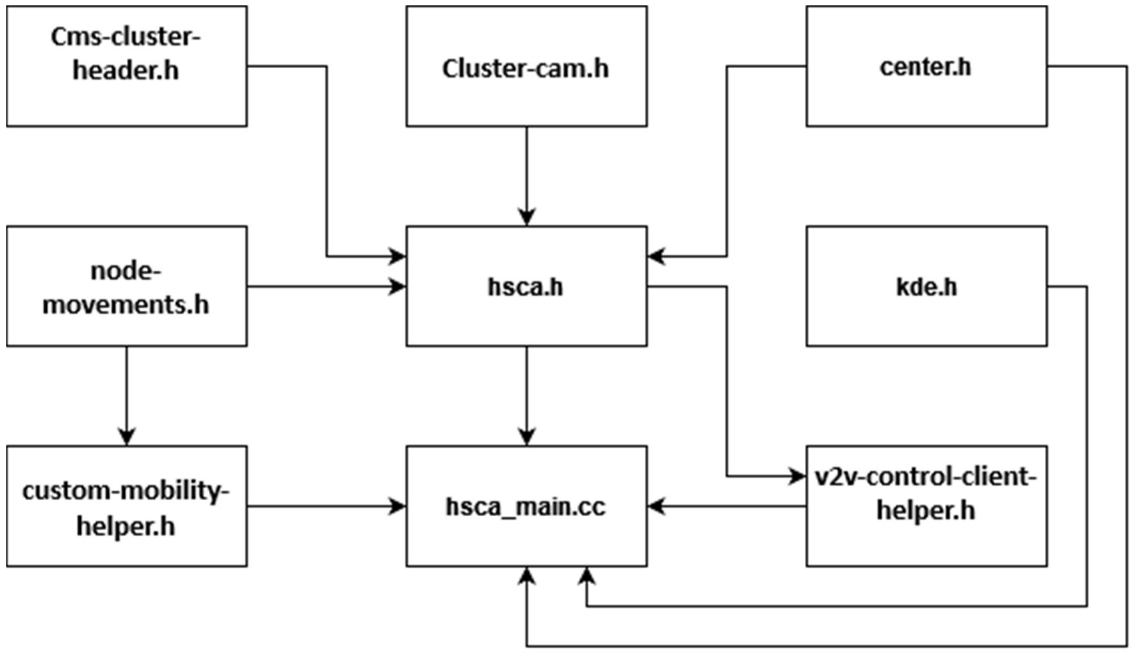


Figure 3.7: Files that were implemented in order to simulate the HSCA algorithm and how they were included.

Chapter 4

The Simulation Environment, Results, and the Performance Evaluation

In the previous chapter three based mobility clustering algorithms have been discussed. So, in this chapter, we will present the technologies used to simulate those algorithms that will be simulated and discussed with several experiments that will be performed, with many different simulation parameters and values that may affect the simulation results (quality of clustering algorithm).

4.1 Operating System

Ubuntu is a computer operating system based on the Debian GNU/Linux -distribution. Linux is a general name for a UNIX-like computer operating system based on the Linux kernel and gathered for libraries and system programs developed within the limits of the project GNU. Their development is one example of free and open-source software collaboration. This means that all the underlying source code can be used for freely viewed, modified and redistributed, both commercially and non-commercially, by anyone under the terms of the GNU GPL and other free software licenses. Ubuntu is chosen as the operating system because:

1. No need to acquire a license. Distributions of Ubuntu can be downloaded for free, for example, from the Internet, or order a free CD with it by mail.
2. It is possible to have all the latest updates to the necessary applications that the open-source world has to offer.

3. It is easy to find and install all necessary applications, programming environments.

The version of Linux is Ubuntu 18.04.6 with the code name "bionic".

• Network Simulator Ns-3

NS-3 is a discrete-event network simulator that primarily serves the purpose of simulating Ad-hoc networks. Preceding iterations include ns-1 and ns-2, while the present iteration is known as ns-3. This software is specifically designed to cater to the needs of research and educational endeavors [13].

The development of Ns-3 project started in 2006 and it is open-source. It is implemented in C++ and Python and it is licensed under the GNU GPL (General Public License). The workflow on ns includes the following four steps:

1. Implementation of protocol models.
2. Setup the simulation scenario.
3. Running the simulation.
4. Analysis of the simulation results.

Some basic components of the program are:

1. NS, the simulator itself has Nam, the network animator to visualize ns (or other) output.
2. Pre-processing component for Traffic and topology generators.
3. Post-processing for Simple trace analysis (in Awk, Perl or Tcl).

• NS3 Basics [13]

The NS3 network simulator is implemented in the C++ programming language and employs Python scripts for the compilation and execution of code. Simulations are primarily written in C++ and the waf build tool [12], which is written in Python, is utilized for code compilation and execution. While it is feasible to write simulation scripts in Python, it should be noted that due to the prevalent usage of C++ among users, obtaining assistance from other users may not be as readily available.

• Simulation Components

To understand how NS3 simulation works, we need to understand how a simulation is built. The main components we need to get started with are:

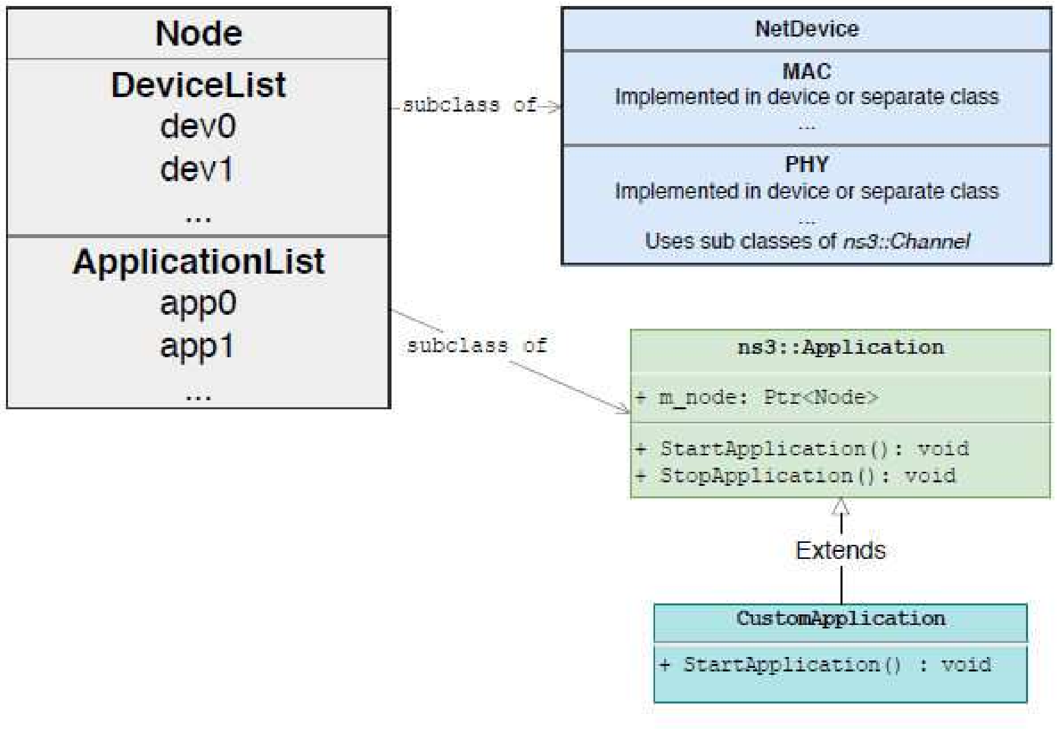
1. **Nodes:** A node represents a network host. It is defined as a type named **ns3::Node**. When a node is created, it is created without a position. A node's position is only

needed if it uses a wireless communication device since that's how signal propagation is calculated.

The main components of a nodes are:

a. **Net devices:**
A node has the capability to contain a list of net devices, which is a subclass of ns3::NetDevice. Net devices possess the ability to access the pointer to their respective node. This implies that if an object of a net device is pointed to, the node in which it is contained can be accessed by utilizing the GetNode function. Net devices may contain the implementation of PHY & MAC within the same class, such as CsmaNetDevice, or they may have PHY & MAC implemented in separate classes, as in WifiNetDevice.

b. **A list of applications**
A node has the capability to store a collection of applications, which is implemented as a subclass of ns3::Application. Users have the ability to define the start and stop times for these applications. Similar to network devices, applications can retrieve their associated node using the **GetNode** method, thereby gaining access to the node's network devices as well.



Node
DeviceList
dev0
dev1
...
ApplicationList
app0
app1
...
subclass of
NetDevice
MAC
Implemented in device or separate class
...
PHY
Implemented in device or separate class
...
Uses sub classes of ns3::Channel
subclass of
ns3::Application
+ m_node: Ptr<Node>
+ StartApplication(): void
+ StopApplication(): void
Extends
CustomApplication
+ StartApplication() : void

Figure 4.1: The main components of nodes

2. **Channels:**
   In order to facilitate the transmission of data between devices, it is necessary to establish a channel object, which serves as a subclass of ns3::Channel. Devices within the same broadcast domain will collectively utilize a single channel object. For instance, in the case of an Ethernet network (where Ethernet devices are implemented in ns3::CsmaNetDevice), all devices within the network will share a common ns3::CsmaChannel object. It is possible to define specific properties for the channel, such as propagation delay and data rate. In the context of WiFi simulation, all nodes will share a channel object of type ns3::YansWifiChannel.

   The characteristics of propagation loss and delay are determined by separate classes, such as ns3::LogDistancePropagationLossModel and ns3::ConstantSpeedPropagationDelayModel. It is important to note that for these models to function accurately, the nodes must have a MobilityModel installed, which enables the specification of wireless signal loss and delay based on the distance between them.
3. **Packets:** Data is exchanged with objects of type Packet. In ns-3, IP segments and data-link frames are also considered as packets. Headers can be added or removed from a packet, thereby altering its size. Additionally, packet tags can be utilized to attach supplementary information to a packet without modifying its content. Further details regarding packet tags will be discussed later in this document.
4. **Event:**

   If one desires a certain action to occur after a specified duration, it is necessary to arrange an event that executes a function. This can be achieved by utilizing the function's pointer (using the "&" symbol) and providing any relevant parameters to be passed to said function (if applicable) (see Appendix 1.A).
   The following demonstrates how one can schedule the execution of the function "DoSomething" to occur after a 3-second interval. Additionally, it includes the passing of arguments "x" and "y". It should be noted that this example pertains to a straightforward scenario where functions are not encapsulated within a C++ class.

   There may arise situations where it becomes imperative to execute certain code periodically. For instance, one may wish to periodically display simulation data or write to a file every second. It is indeed possible to create a function that schedules a recurring call to itself every second. It is crucial to remember to schedule the initial call to the function as well (see Appendix 1.B).

5. **Smart Pointers and ns3::Object**

   In the C++ programming language, it is possible to create a pointer to an object by utilizing the * symbol and the new keyword, as demonstrated in the preceding section. The ns-3 framework incorporates the concept of smart pointers, provided that the C++ classes are derived from the ns3::Object class. It is worth noting that the Ptr class template is employed in conjunction with the aforementioned classes. This is the approach employed by ns-3 to implement smart pointers, which is similar to the boost library. The majority of classes within ns-3 utilize Ptr<T>, where T represents the type name, to generate pointers to objects. This approach offers various functionalities, including type checking, casting, and invoking constructors. When it comes to object creation, instances of an ns-3 class that is a descendant of ns3::Object can be created using the CreateObject function. For instance, to create a single node (ns3::Node), the following code can be utilized (refer to Appendix 2.A).

This effectively invokes the default constructor of the Node class, which does not require any parameters. It is important to bear in mind that, in most cases, it is preferable to create nodes using the NodeContainer class (refer to Appendix 2.B).

It is worth mentioning that certain classes within NS-3 are subclasses of ns3::SimpleRefCount<T>, rather than ns3::Object. An example of a class falling into this category is the Packet class, and objects of this type can be created using the Create function (refer to Appendix 2.C).

A. Type checking and casting:

C++ is an object-oriented programming language that has inheritance of classes. In ns-3, we have many abstract classes that are extended throughout ns-3. For example, ns3::NetDevice is an abstract class because it has a pure virtual function Send. Some of the subclasses of NetDevice are WifiNetDevice, CsmaNetDevice and many others. You will see in the examples that we have a NetDeviceContainer, which contains pointers to objects of types extending NetDevice, and ApplicationContainer which contains pointers to objects of types extending the ns3::Application class.

We can examine the run-time type of an ns-3 object by using the function GetInstanceTypeId(), which returns a value of type ns3::TypeId. In addition, most ns-3 classes are designed with a static function that returns the TypeId of a class.

Recall that nodes in ns-3 can have multiple net devices. The (Appendix 3.A) iterates through the net devices of a node, and perform type casting on them.

Similar casting can be done with objects in ApplicationContainer, as ns-3 applications are subclasses of ns3::Application (Appendix 3.B).

If you need to create a C++ class that is compatible with ns-3 smart pointer functionalities, you need to extend the ns3::Object and create the functions GetInstanceTypeId and GetTypeId.

B. Object Aggregation

In ns-3, an object of a type that is a subclass of ns3::Object can be associated with another object that is a subclass of ns3::Object using the AggregateObject. You can only aggregate a single object of a given type to another object. When two objects are aggregated, you can use the function GetObject on one object to retrieve a pointer to the other.

For example, if you have two classes ClassOne and ClassTwo that are both subclasses of ns3::Object, you can perform (Appendix 4.A).

Object aggregate is used to implement mobility in ns-3. Nodes created in ns-3 have no mobility information, so no position or velocity. Nodes without mobility can only be used in wired communication because it does not need position information to determine reception. You do not need to use AggregateObject yourself to give nodes position, but you may need to use GetObject to determine the mobility information of a node (Appendix 4.B).

This is convenient because all simulation nodes are contained in the global NodeList container. So, we can inquire about nodes positions anywhere in our code. In addition, we may aggregate objects of our own user-defined classes to nodes (like center class that will be explained later), and have that information available anywhere in the code.

• Eclipse

For the sake of convenience in code usage and spelling, We opted to utilize Eclipse. Eclipse is a comprehensive software development environment that encompasses an integrated development environment (IDE) and an expandable plug-in system. It is primarily coded in Java and can be employed for the creation of applications in Java, as well as other languages such as C, C++, COBOL, Python, Perl, PHP, and more, thanks to its diverse range of plug-ins. specifically employed the Eclipse CDT module for C/C++ codes, as ns-3 is implemented in C++. Additionally, Eclipse includes a highly advantageous debugging tool. Debugging is a systematic process aimed at identifying and minimizing the number of bugs or defects present in a computer program.

• Installation:

Prior to beginning the installation of Eclipse, it is imperative to install the sun-java6-jre package with the assistance of the package manager. This manager can be utilized to install the Eclipse IDE, and subsequently, through the Eclipse interface, the Eclipse C/C++ Development Tools (CDT) and other plugins can be installed to facilitate a more convenient working environment. Alternatively, one can simply download the "Eclipse IDE for C/C++ Developers" distribution from the official website at http://eclipse.org/.

Before proceeding with the installation of ns-3, it is necessary to prepare the system accordingly. The ns-3 core necessitates the presence of a gcc/g++ installation of version 3.4 or higher, as well as python 2.4 or a more recent version. Detailed instructions for the proper installation of all the prerequisites for ns-3 can be found on the webpage http:://www.nsnam.org/wiki/index.php/Installation. Additionally, on this page, instructions for the installation of ns-3 itself can be found, although it is advisable to download the latest version from the official site at http://www.nsnam.org/.

- **Configure ns-3 in Eclipse:**

When initiating Eclipse, it is advisable to modify the workspace to the directory that houses the ns-3 files (e.g., home/username/repos). Proceed to create a new empty C/C++ project with a name identical to the folder name of the ns-3 project (e.g., "ns3.7"). Once the creation of the empty project is complete, it will automatically populate with the necessary ns-3 files. This process ensures a convenient working environment for ns-3 files.

• SUMO

The acronym SUMO represents Simulation of Urban Mobility and is governed by the GPL (GNU General Public License). It is an open source software that is utilized for the development and simulation of highly portable, microscopic, and continuous road traffic packages that are specifically designed to manage extensive road networks. In greater detail, a network file generated by SUMO has the capability to describe the traffic associated with a particular section of a map. This file comprises the roads of the network, including intersections or junctions, as well as the traffic lights of the map. The file format is XML (eXtensible Markup Language), but it is not intended to be manually edited. There are certain programs (NETGEN, NETCONVERT) that are associated with SUMO and facilitate the creation of networks in a user-friendly manner. Additionally, SUMO provides a diverse range of extensions to enable efficient communication with other programs, such as ns3, which is essential for our simulation.

- **The format of the network**

In a conceptual sense, it can be stated that a SUMO network can be characterized as a directed graph wherein the roads serve as the edges and the intersections/junctions act as the nodes. To delve deeper into the matter, additional aspects of traffic are associated with the following details:

1. Each street/road comprises multiple lanes.

2. Specific attributes such as position, shape, and speed limit can be attributed to each lane.

3. The right-of-way regulations are taken into consideration.

4. The connections between lanes, which correspond to junctions/intersections, are accounted for.

5. The placement and operational logic of the traffic lights are also considered.

- **The Simulation Network**

Having explored the basic components of both SUMO and NS3, we can move on to the presentation of the topologies of the network that were used for the simulation.

The topology simulates the movement of 560 nodes in a highway. The topology consists of a highway scenario of 8 lanes, 4 in each direction is implemented. The highway section is 6 km long and an additional side exit is placed at the 3 km mark for both directions, as seen in Figure 5-2. The side exit is accessible only to vehicles driving on or that move to (due to the lane changing model) the side lane. The probability that a vehicle on the side lane leaves the network at the side exit is p = 0:7.

During the simulation, they were constructed 100 nodes – vehicles of same type (car). The vehicles varied slightly in the maximum velocity that they could move and the minimum gap that represents the minimum distance that they have to keep from the car that moves in the front. The details about the movement of the vehicles were declared in the "highwar_rou.xml" file.

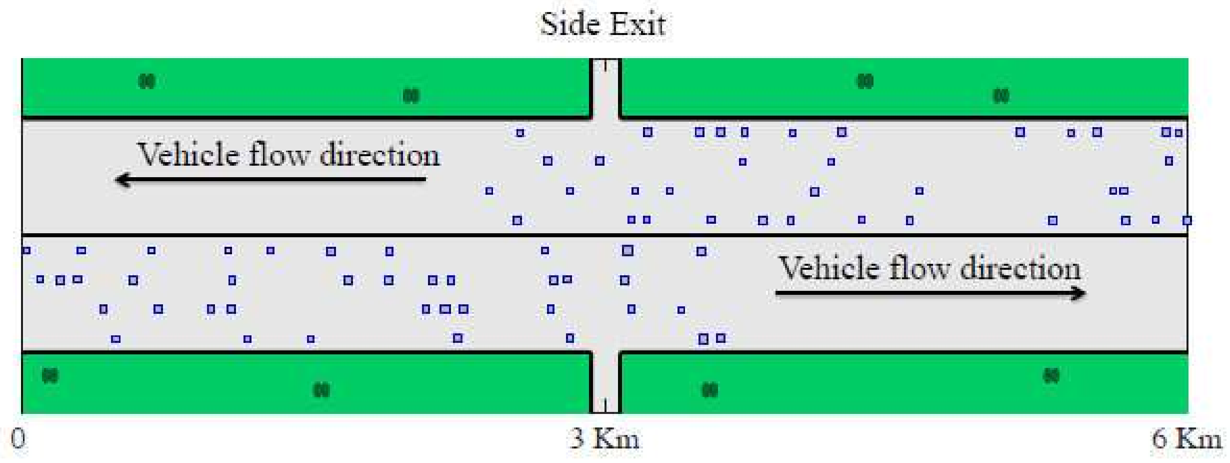


Figure 4.2: Simulations scenario

Highway of four lanes for each moving direction. Length of road section is 6Km and 2 side exits are placed at 3Km to allow vehicles driving on or moving to the side lane to leave the highway. Arrival rate of cars = 5 car/min for illustrative purpose only.

4.2 Initial files and description

To streamline the code implementation process for all algorithms, initial files were generated to read the network that is prepared for simulation and to load the position details of each node into memory. These files, which exhibit near-identical characteristics across all algorithms in (Appendix 5.A).

If the user does not provide an adequate number of arguments, the program is terminated by providing the user the message "Please enter the number of nodes and the period of times that the program runs as the second and the third parameter. Also, give the file name that declares the movement and log file name. This file creates nodes, assign its movements, sets the transmission radio of the nodes that participate in the simulation, and install algorithms applications to these nodes.

Finally, at the end of the file, there is a call to the main function of each of the three algorithms that were implemented (Appendix 5.B).

4.3 Theoretical Comparison of CBSC, SCalE, hybrid algorithm

Depending on the theoretical goals of each of the mentioned algorithms and the ways to achieve them, we are supposed to get the following results:

Table 4.1: Theoretical comparison of CBSC, SCalE, hybrid algorithm.

Algorithm	**Parameters for CH election**	**Data Routing**	**Index**	**Advantages**	**Disadvantages**
CBSC	Speed difference (V_k), acceleration Difference (A_k).	Multi hop Routing	Relative Mobility Metric (M)	-Improve the reliability and efficiency of communications. -Able to outperform SCalE average CH life time, packet loss rate and average CM life time.	-Require significant infrastructure investment to install GPS devices and 802.11p and LTE interfaces in every vehicle, which can be costly and time-consuming. -It fails to outperform ScalE in Average CH life time.

SCalE	Vehicle Behavior (Bk), Relative Mean Speed (Sk), Mean relative distance (Dk)	From Cluster Head to UAV	Lowest Index (ξk)	Enhance the cluster stability in average CH lifetime and reduce data loss rate.	-It need a complex infrastructure in order to be able to get all the requirements of this algorithm. -Compared to CBSC the average Life Time of CM was lower.
HSCA	Vehicle Behavior (Bk), Relative Mean Speed (Sk), Mean Relative distance (Dk	Multi hop Routing	Lowest Index (ξk)	-Improve the reliability and efficiency of communications. -Enhance the cluster stability in average CH lifetime and reduce data loss rate.	Require significant infrastructure investment to install GPS devices and 802.11p and LTE interfaces in every vehicle, which can be costly and time-consuming.

4.4 Comparison of Algorithms for Stable Clustering in VANETs

4.4.1 Simulation Parameters:

Since the proposed algorithm depends on the vehicles mobility, which includes speed , position and direction, the simulation parameters must be related to that, so we chose different values for these parameters, and since VANET networks depend on communication between vehicles through Wi-Fi technology, which represents the cluster domain, therefore we must choose Different values of the communication field between the vehicles and its head, and conduct simulations of these values on a different number of vehicles .

Table 4.2: Baseline parameters used in the simulations

Parameter	Value
Simulator version	NS3
Mobility Model	ConstantVelocityModel
MAC layer protocol	IEEE 802.11
Antenna Model	Omni-Antenna
Channel type	Wireless
Simulation area	6 km highway in both direction
Maximum Speed of nodes	0.001, 0.01, 0.03 (Km/S)
Packet size	512 (byte)
Simulation time	4032 s
Cluster range	100,200,300,450 (m).
Vehicles Number	250 vehicles

4.4.2 Simulation comparison of CBSC, SCalE, hybrid algorithm

Since the proposed algorithm depends on the vehicles mobility, which includes speed, position and direction, the simulation parameters must be related to that, so we chose different values for these parameters, and since VANET networks depend on communication between vehicles through Wi-Fi technology, which represents the cluster domain, therefore we must choose Different values of the communication field between the vehicles and its head, and conduct simulations of these values on a different number of vehicles.

Several experiments were performed, and the results were taken at different numbers of nodes, different mobility speed for nodes and different range of cluster.

- **Effect of numbers of nodes:**

This section shows the results from our simulation experiments when varying the number of nodes (vehicles), between 100 nodes,200 nodes, 450 nodes and 560 nodes.

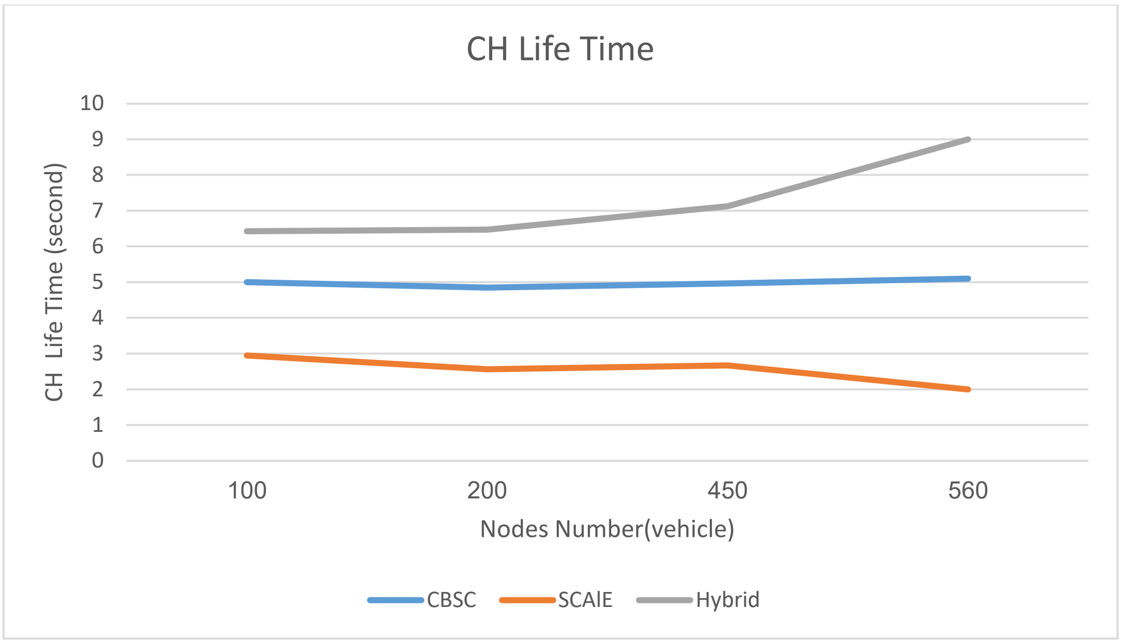


Figure 4.3: Effect of number of nodes on CH Life Time

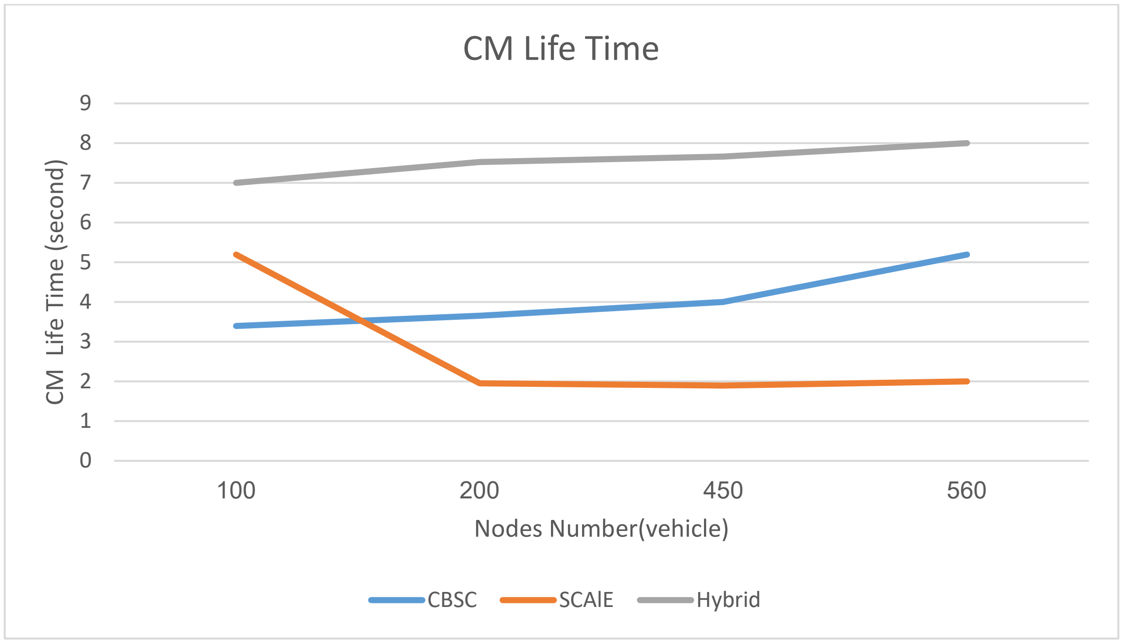


Figure 4.4: Effect of number of nodes on CM Life Time

Figures 4.3 and 4.4 show the Effect of number of nodes on CH life time.

In CBSC Algorithm, an increase in the number of nodes leads to an increase of cluster head life time and Cluster member life time, this is due to the increase in the size of the cluster and the density of the distribution of nodes, the clusters centers will be as close as possible to all points and when electing the most appropriate one, we have elected the best node among the largest possible number of nodes.

In SCalE Algorithm, any increase in the number of nodes leads to a little increase of cluster head life time and Cluster member life time, this is due to the increase of the number of adjacent nodes, But the centers of the clusters, represented by the coordinates of the cluster header's position, quickly change, so there will be a large number of cluster head and cluster members state changes with each increase in the number of nodes.

In HSCA Algorithm   , as CBSC algorithm , after locating the centers of clusters, the number of clusters always remains constant, while the number of its elements changes, as if the cluster   is a fixed circular patch that vehicles pass through, and thus a specific number of state changes for the elements of this cluster, as it moves from one cluster to another until reaching the target position , so it will be There is a limited number of times the vehicle's state changes, and these vehicles will remain within this cluster from the time they enter the cluster range until they exit it, so a long life time for the cluster members and cluster head.

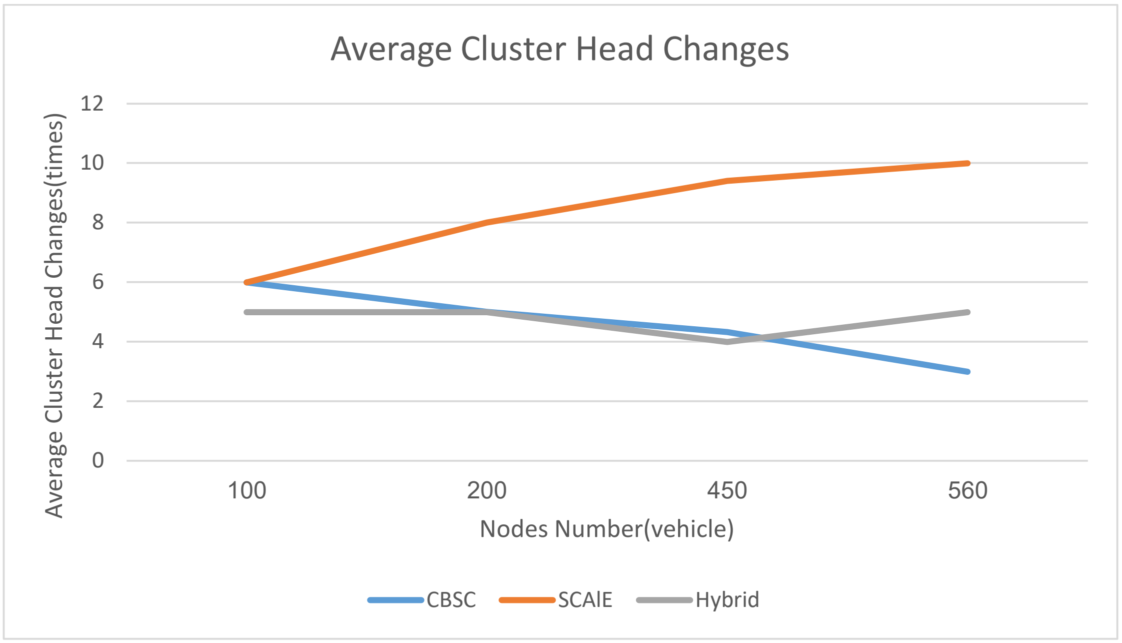


Figure 4.5: Effect of number of nodes on average cluster head changes

Figure 4.5 shows the effect of number of nodes on average cluster head changes.

In CBSC Algorithm, we have elected the best node among the largest possible number of nodes. Therefore, cluster head re-election will be limited to cluster head leaving the cluster and therefore few state changes.

In SCalE Algorithm, the centers of the clusters, represented by the coordinates of the cluster header's position, quickly change, so there will be a large number of cluster head and cluster members state changes with each increase in the number of nodes.

In HSCA Algorithm, vehicles will remain within this cluster from the time they enter the cluster range until they exit, so it will be There is a limited number of times the vehicle's state changes.

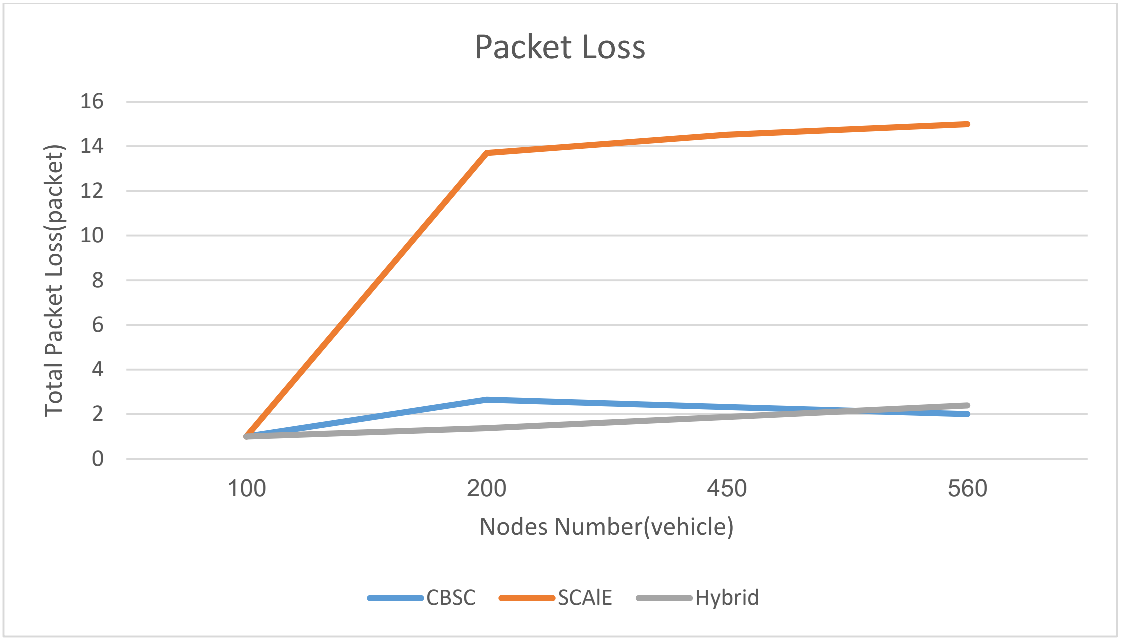


Figure 4.6: Effect of number of nodes on packet loss

Figure 4.6 show the Effect of number of nodes on packet loss.

In CBSC Algorithm, long CH and CM life time cause Less packets will lose, these cases of packet loss will be limited to cases of nodes leaving the cluster before the packet arrives, and these cases are very few.

In SCalE Algorithm, a high packet loss rate will appear because the high rate of Nodes degree changes (CM-CH).

In HSCA Algorithm, a non-existent loss of packages due to the long time of the vehicle within the cluster, and less status changes, so long connection time with neighbor’s vehicles.

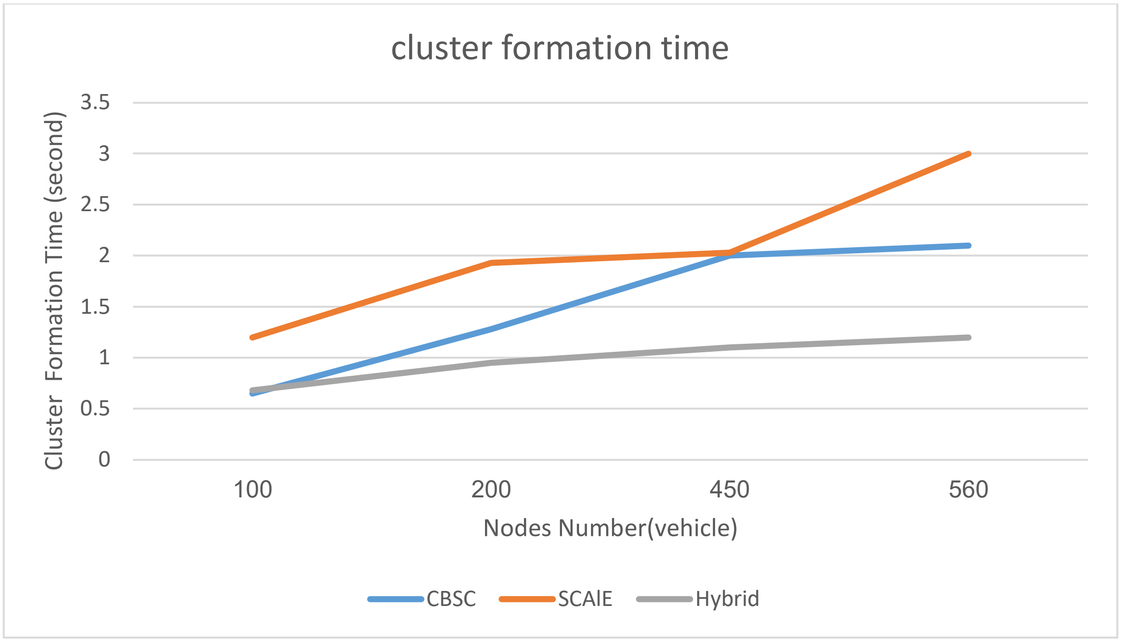


Figure 4.7: Effect of number of nodes on cluster formation time

Figure 4.7 shows the Effect of number of nodes on cluster formation time.

In CBSC Algorithm, cluster formation time will be increased by the increase of the number of nodes, due to cluster head suitability metric calculation time for all nodes.

In SCalE Algorithm, cluster formation time will take long because of the large number of messages exchanged between nodes.

In HSCA Algorithm, cluster formation will be as fast as possible, this is due to the increase in the size of the cluster and the density of the distribution of nodes, Once the cluster formation packet arrives, the cluster is formed.

- **Effect of cluster range:**

This section shows the results from our simulation experiments when varying the cluster range, between 80m ,150m, 200m and 250 m.

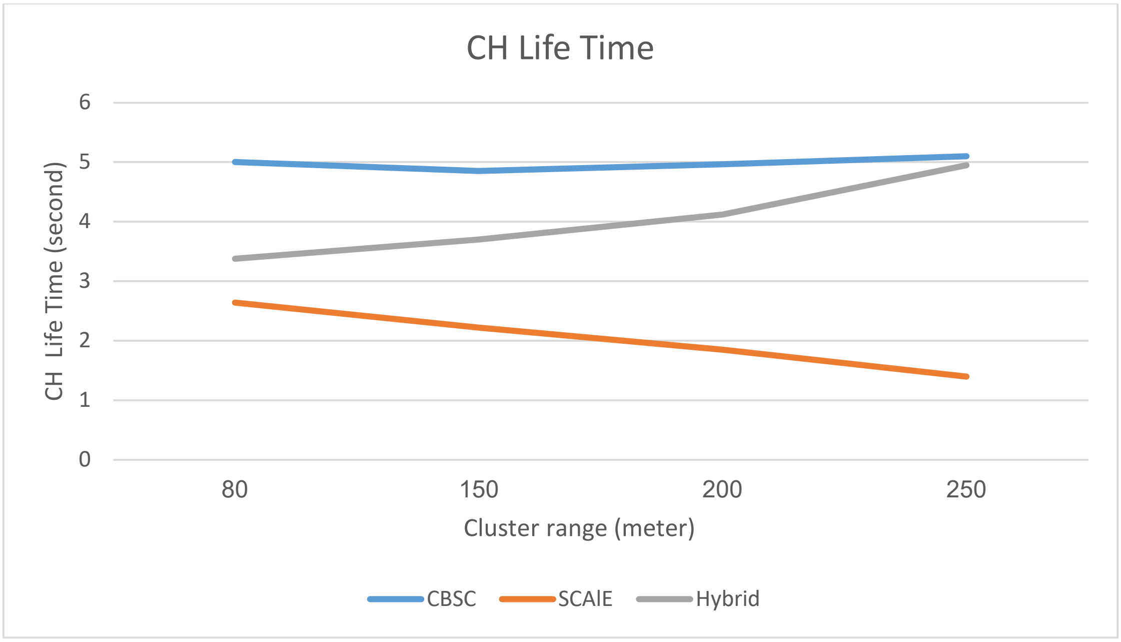


Figure 4.8: Effect of cluster range on CH Life Time

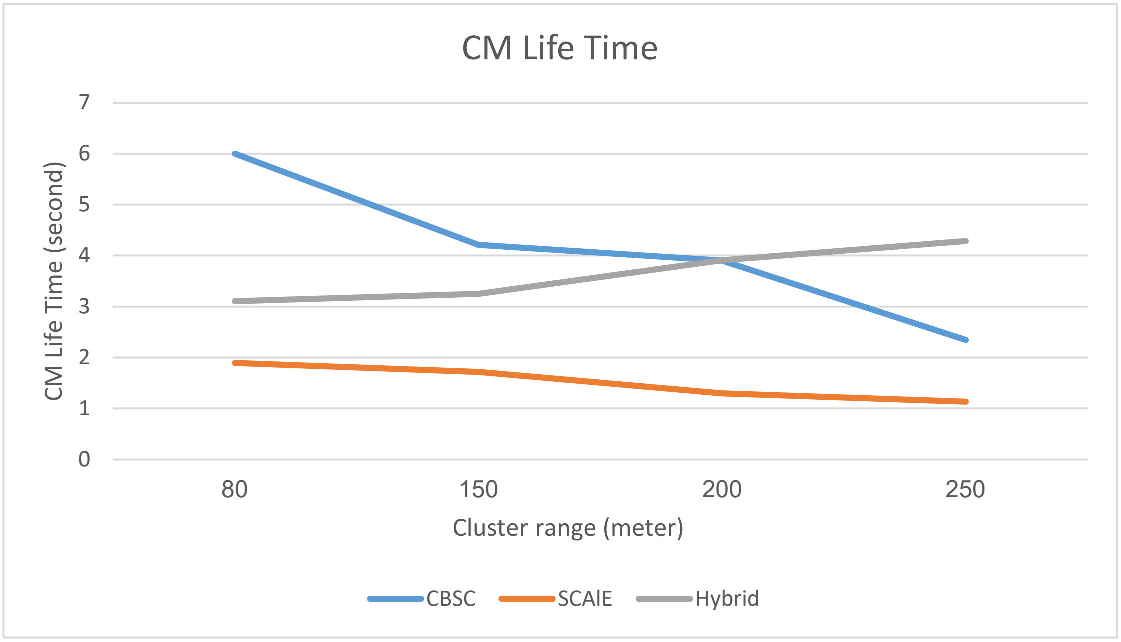


Figure 4.9: Effect of cluster range on CM Life Time

Figure 4.8 and 4.9 show the Effect of cluster range on CH life time and CM life time respectively.

In CBSC Algorithm, an increase in the cluster range leads to an increase of cluster head life time and Cluster member life time, this is due to the increase in number of nodes that covered by cluster.

In SCalE Algorithm, increasing cluster range cause less cluster head and cluster member life time, because the centers of the clusters, represented by the coordinates of the cluster header's position, quickly change, so there will be a large number of cluster head and cluster members state changes.

As for HSCA Algorithm, any increase of cluster range means wide coverage area, which will be more useful to increase cluster head and member's life time.

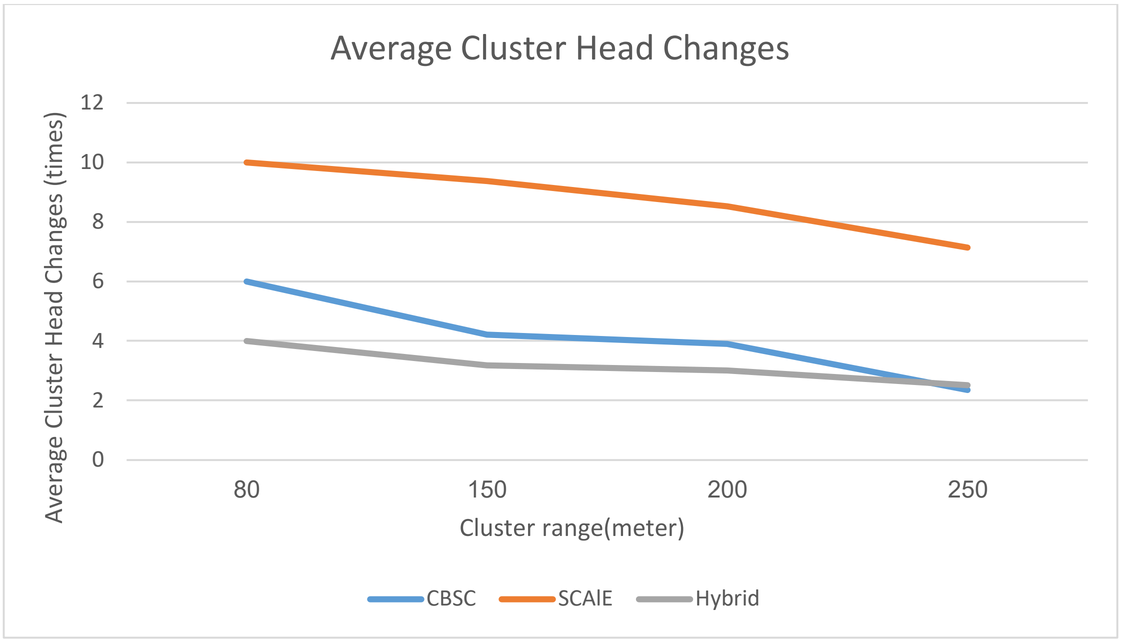


Figure 4.10: Effect of cluster range on average cluster head changes

In CBSC Algorithm, an increase in the cluster range leads to an increase in number of nodes that covered by cluster, and will stay for as long as possible at this cluster, which means less cluster head changes.

In SCalE Algorithm, increasing cluster range cause large number of cluster head changes will because none fixed cluster's centers.

In HSCA Algorithm, any increase of cluster range means wide coverage area of nodes, so less cluster head changes.

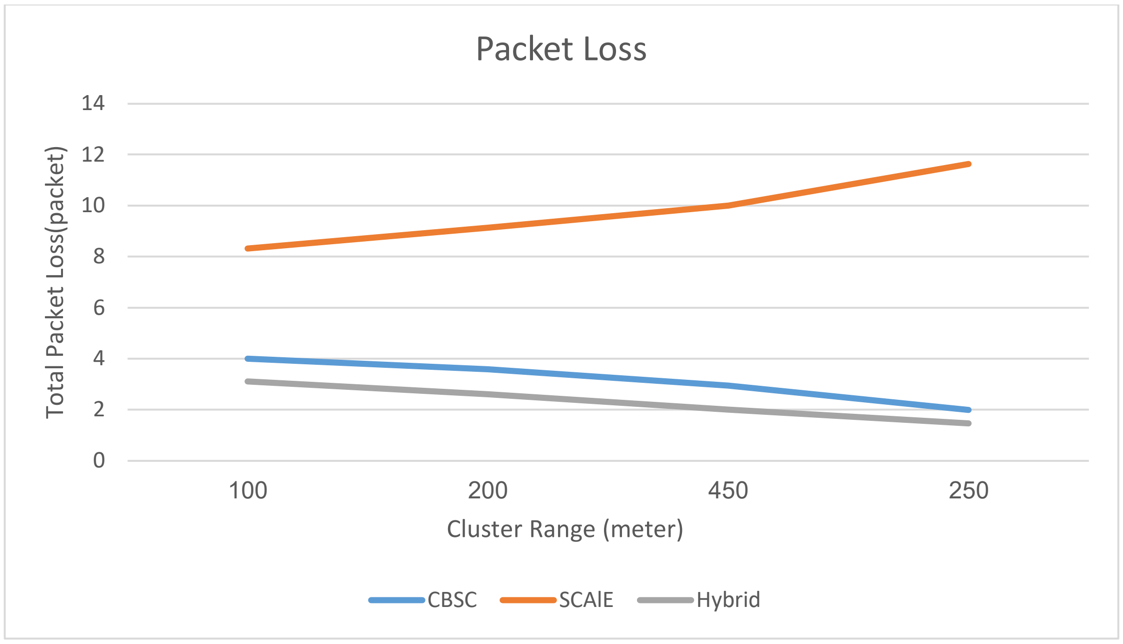


Figure 4.11: Effect of cluster range on average cluster head changes

As Figure 4.11 shows:

In CBSC Algorithm, an increase in the cluster range leads to more stable in cluster connections and less packet loss.

In SCalE Algorithm, increasing cluster range means less cluster head and cluster member life time, so, high packet loss.

In HSCA Algorithm, any increase of cluster range means more stable in network and minimal packet loss.

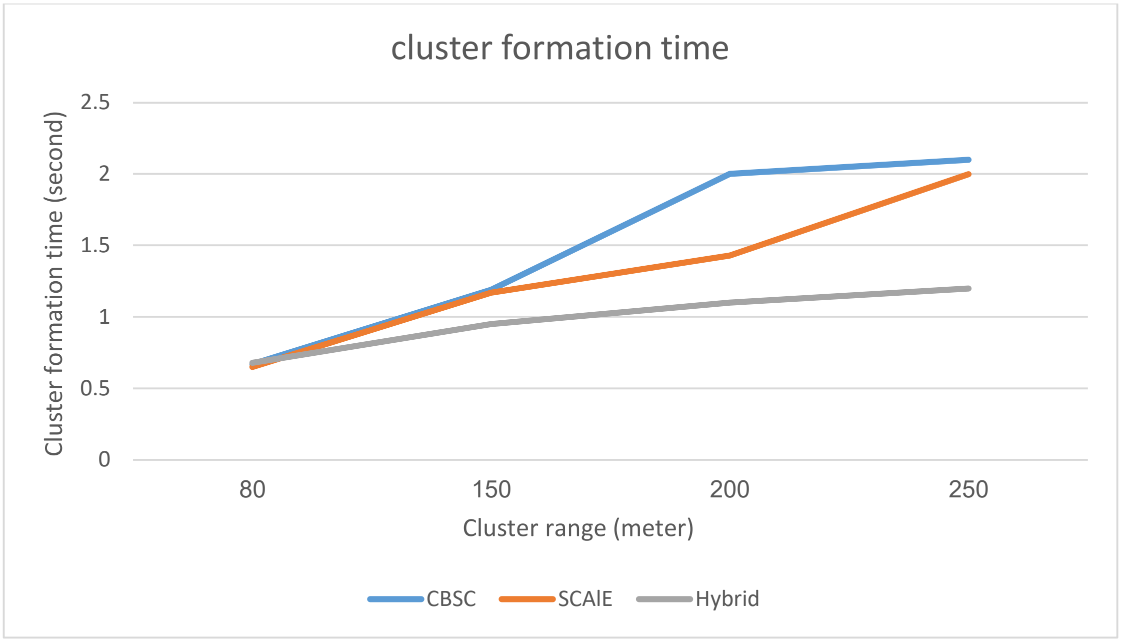


Figure 4.12: Effect of cluster range on average cluster formation time

In CBSC Algorithm, Large cluster range means more and more nodes with the same cluster, so more calculation and cluster formation time.

In SCalE Algorithm, any loss of connection means more time to cluster formation or maintenance, Large number of packet should be waited before cluster formation.

In HSCA Algorithm, once the cluster formation packet arrives, the cluster is formed.

- **Effect of mobility speed:**

This section shows the results from our simulation experiments when varying the vehicles mobility speed, between 5km/s ,15km/s, 20km/s and 30 km/s.

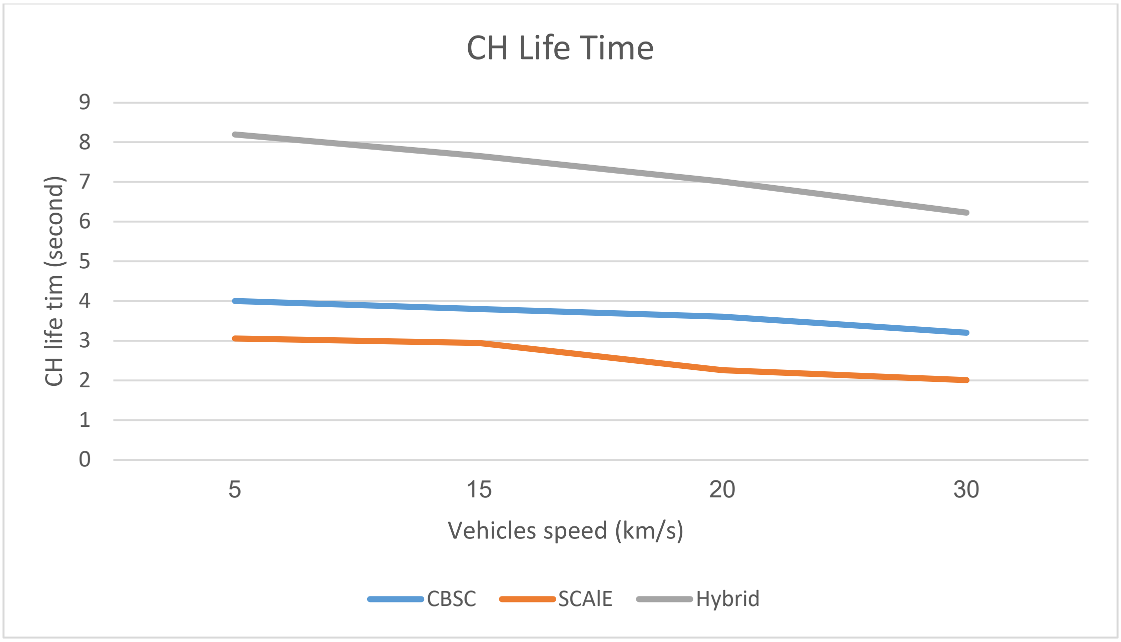


Figure 4.13: Effect of mobility speed on CH Life Time

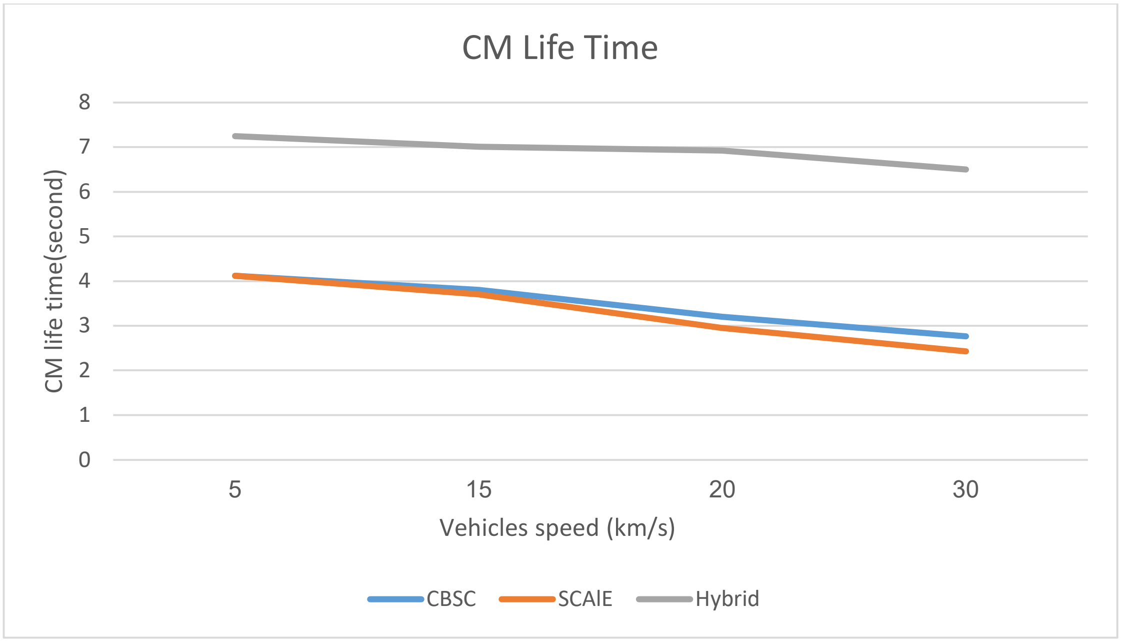


Figure 4.14: Effect of mobility speed on CM Life Time

Figures 4.13 and 4.14 show the Effect of mobility speed on CH Life Time and CM life time respectively.

In CBSC Algorithm, as shown at this figure, the Varity of speed will decrease cluster head and cluster member's life time, because it will leave its cluster after shortest time due to its high speed.

In SCalE algorithm, any increase in mobility speed will cause less stability in cluster, less cluster head and members life time, each node will move far and far from other node (cluster head or member.

In HSCA, For the same reason in CBSC algorithm, high speed cause quick departure from the cluster, but it will stay at cluster as long as it's at cluster range, so cluster head and member life time dependent on cluster range when nodes move with high speed.

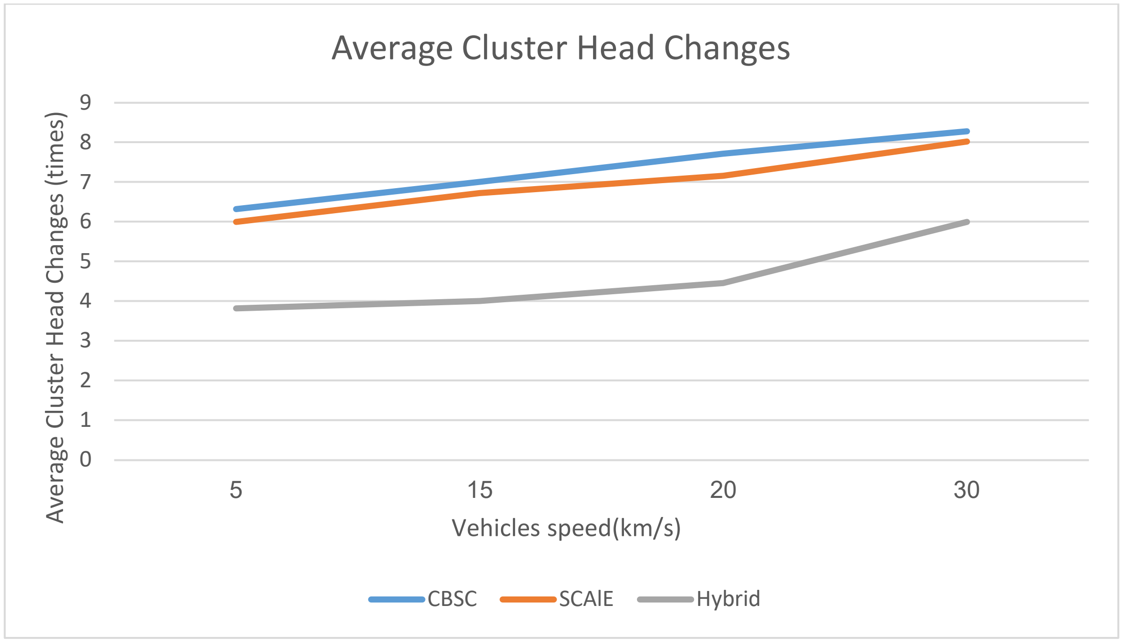


Figure 4.15: Effect of mobility speed on CH changes Life Time

In CBSC Algorithm, as shown at this figure, the Varity of speed will decrease cluster head and cluster member's life time, so will cause more number of cluster head changes.

In SCalE algorithm, as shown in figure 4.15, any increase in mobility speed will cause less stability in cluster, and more cluster head changes.

In HSCA Algorithm, the number of cluster head changes dependent on cluster range.

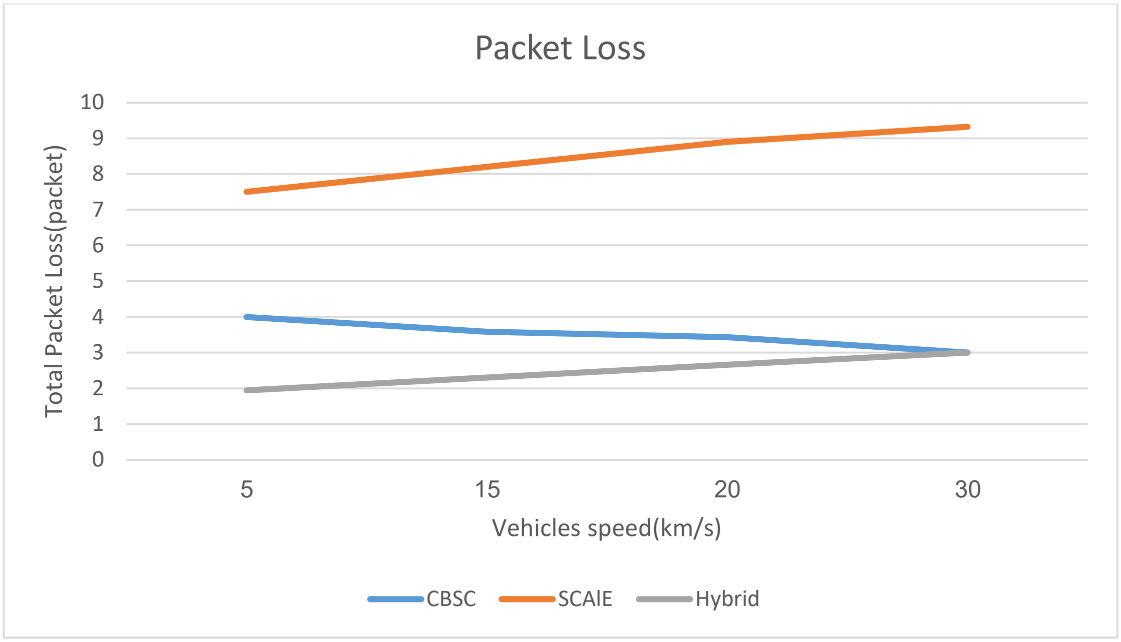


Figure 4.16: Effect of mobility speed on Packet Loss

In CBSC and HSCA, the packet loss rate dependent on cluster range, so at high speed, large cluster range means less packet loss and vice versa.

In SCalE, high speed means high packet loss rate and connection failures.

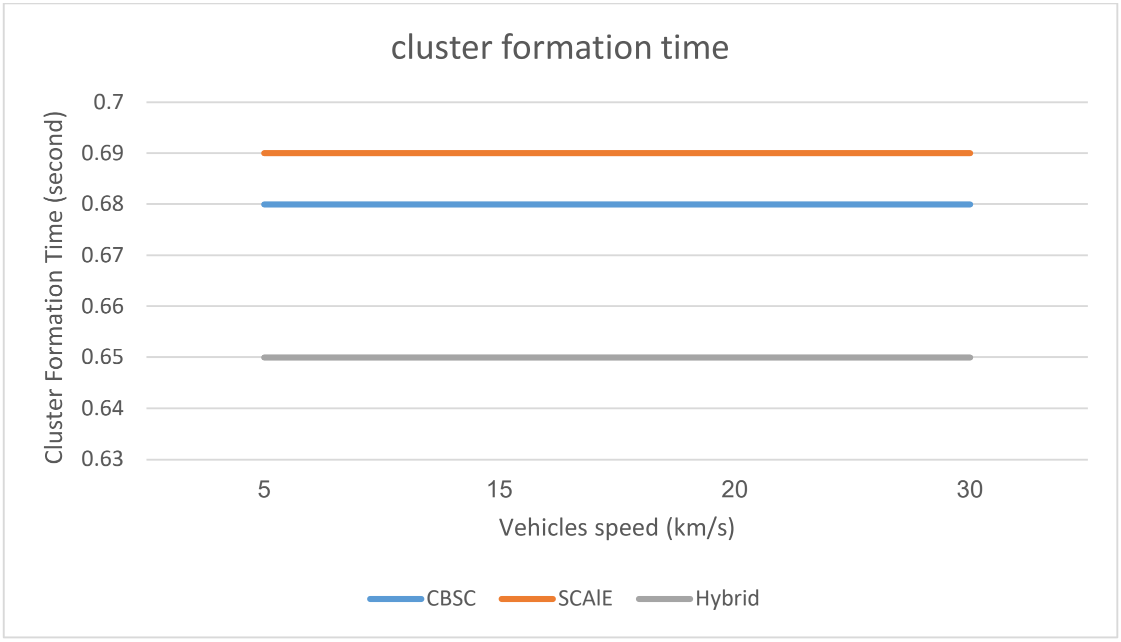


Figure 4.17: Effect of mobility speed on cluster formation time

As shown at above figure, the Varity of speed does not have any effect on cluster formation time at any algorithm.

- **Effect of mobility speed randomization:**

This section shows the results from our simulation experiments when the vehicles mobility speed is random speed between 1km/s and 50km/s.

In this experiment, we fixed the number of nodes at 250 nodes, the cluster range was 150 meters, and the speed was random for each vehicle. The results were the following:

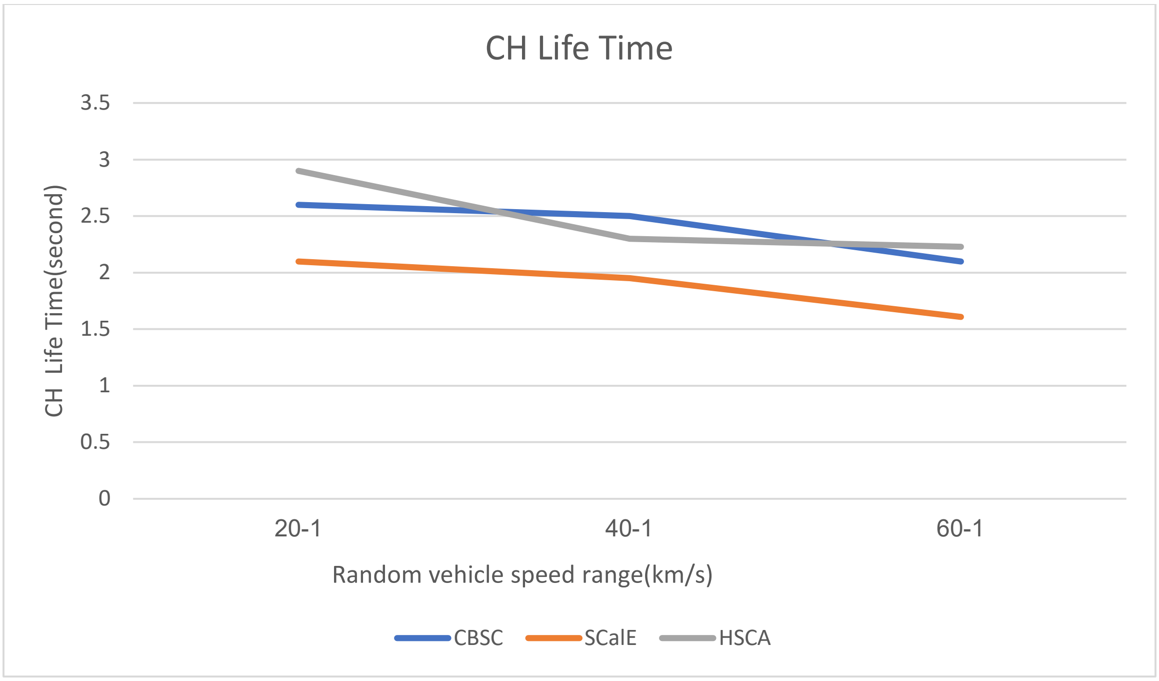


Figure 4.18: Effect of random mobility speed on CH Life Time

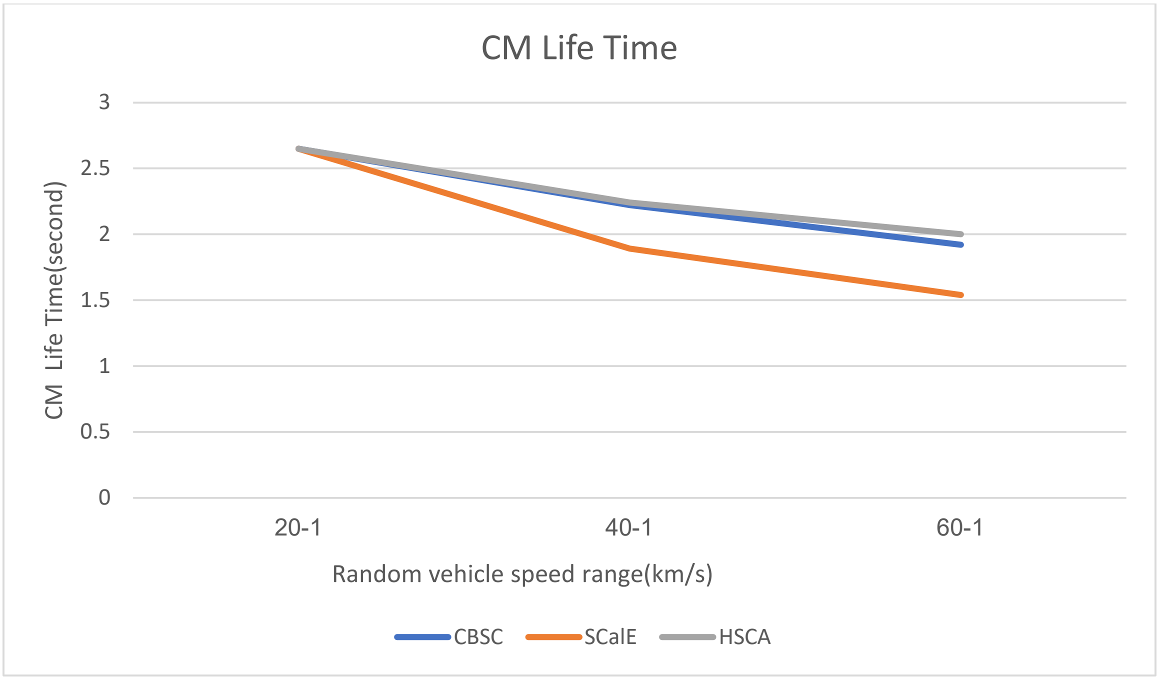


Figure 4.19: Effect of random mobility speed on CM Life Time

Figures 4.18 and 4.19 show the Effect of random mobility speed on CH Life Time and CM life time respectively.

In CBSC Algorithm, as shown at this figure, the random of speed will random cluster head and cluster member's life time, because it will leave its cluster after random time.

In SCalE algorithm, any change in mobility speed will cause change stability in cluster, less cluster head and members life time.
In HSCA, For the same reason in CBSC algorithm, random speed cause random departure from the cluster, but it will stay at cluster as long as it's at cluster range, so cluster head and member life time dependent on cluster range when nodes move with random speed.

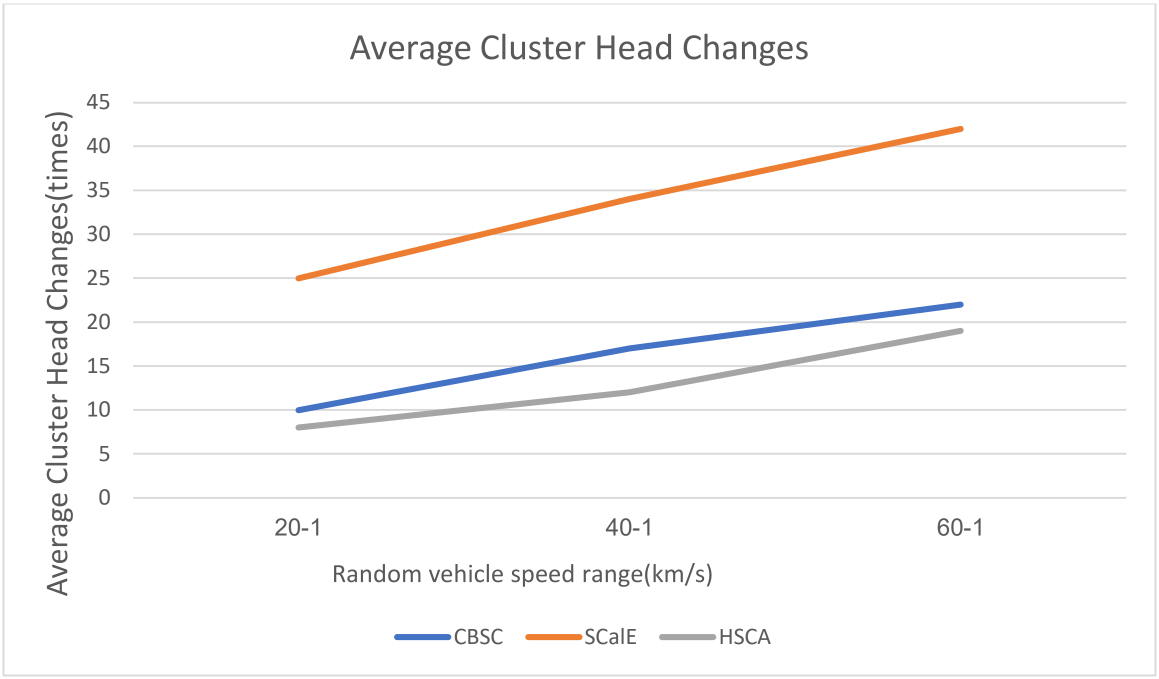


Figure 4.20: Effect of random mobility speed on CH changes

In CBSC Algorithm, as shown at this figure, the random of speed will random cluster head and cluster member's life time, so will cause random number of cluster head changes. In SCalE algorithm, as shown in figure 4.20, any change in mobility speed will cause less stability in cluster, and random cluster head changes. In HSCA Algorithm, the number of cluster head changes dependent on cluster range.

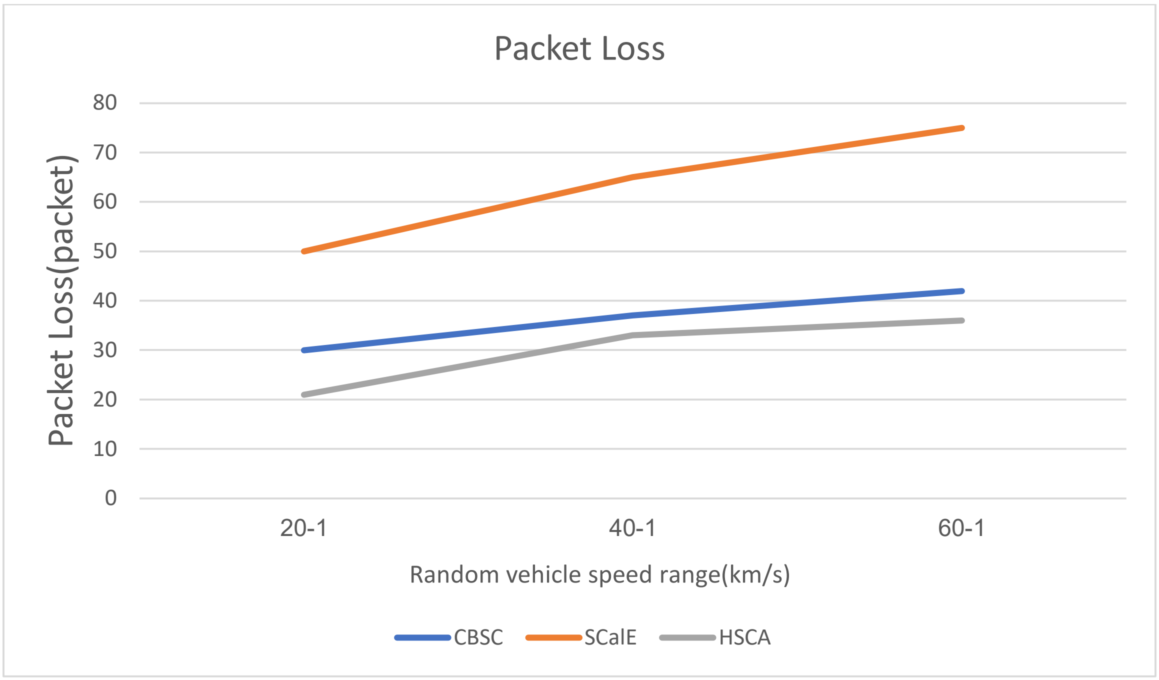


Figure 4.21: Effect of random mobility speed on Packet Loss

In CBSC and HSCA, the packet loss rate dependent on cluster range, so at high speed, large cluster range means less packet loss and vice versa.

In SCalE, random speed means random packet loss rate and connection failures.

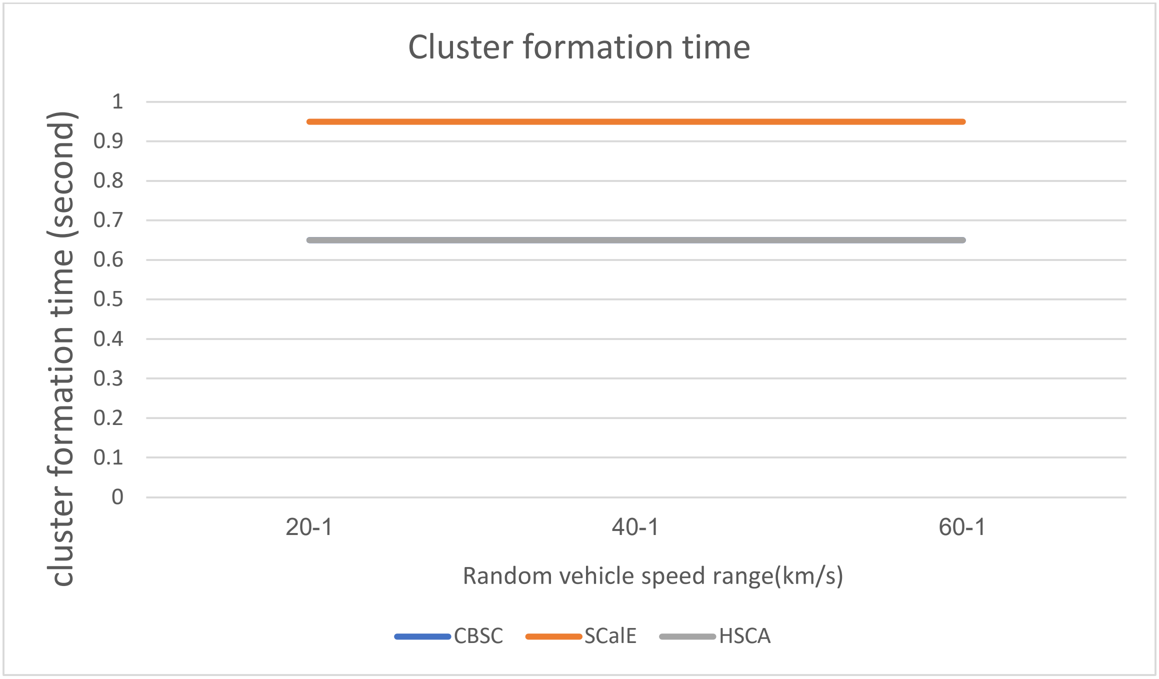


Figure 4.22: Effect of mobility speed on cluster formation time

As shown at Figure 4.22, the random of speed does not have any effect on cluster formation time at any algorithm.

Simulation summary:

From previous experiments, we conclude that any change in the number of vehicles directly affects the results of the algorithm. As the number of vehicles increases, there is a need to cover a larger number of vehicles at varying speeds, in addition to the fact that the area covered by the Wi-Fi communication signal is relatively small given the distances it travels. Vehicles. Therefore, it can be said that in order for stability to be achieved, the speed limits of the vehicles must be restricted so that they remain close to each other and within the cluster range for the largest possible period.

The main factor in increasing the stability of the network is the restricted speed and cluster range.

The percentage of improvement in each case can be calculated through the following equation: % Optimization in HSCA=(((HSCA-CBSC)/CBSC)*100).

Each of the values can be obtained through the graphs shown previously, and the following are some improvement percentages:

The percentage of improvement in the new algorithm (HSCA) in terms of CH life time compared to the previous two algorithms (CBSC and SCalE)in the case of changing the number of nodes was as follows:

Table 4.3 (The percentage of improvement in the new algorithm (HSCA))

# of nodes	CBSC	SCalE	HSCA	%HSCA Vs. CBSC	%HSCA Vs. SCalE
100	5	2.95	6.43	28.6%	117.9%
200	4.85	2.56	6.47	33.4%	152%
450	4.96	2.67	7.12	43.54%	166.7%
560	5.1	2	9	76%	350%

Average improvement percentage=sum of improvement percentages in each case/4.

Based on the experiments that were conducted, it was found that the proposed algorithm (HSCA) was able to outperform its counterparts (SCalE and CBSC) in the improvement rate. It was able to outperform SCALE in all cases that were tested with a high percentage, and it was also able to outperform the CBSC algorithm in most scenarios, and we clarified Reasons why it is not superior in some scenarios.

The proposed algorithm achieved an improvement rate of 45% in terms of CH life time compared to CBSC when changing the number of nodes, and 196% compared to SCALE.

Likewise, in CM life time, HSCA achieved an improvement of 89% compared to CBSC and 230% improvement compared to SCALE. When implementing the scenario of changing the number of nodes and knowing the impact of this on the time required to form a cluster, the HSCA algorithm achieved a better improvement compared to CBSC and SCALE by 22% and 46.46.85%, respectively. The proposed algorithm was able to outperform SCALE in terms of CH and CM life time under the cluster range change, while the results were close to CBSC, where the improvement rate was 117% and 19%, respectively. The same was true for CM life time, where the percentage was 57% and 2.9%, respectively. While the effect of mobility speed was remarkable, the proposed algorithm achieved an improvement of 187% compared to SCale in terms of CH life time and 116% in terms of CH life time.

Chapter 5

Conclusion and Future Work

This chapter concluded our work on the thesis in Section 5.1. Section 5.2 highlights opportunities for further study.

With the great development in applications supported by (VANETs) and the contributions it has made in the field of driver's assistance, traffic efficiency and road safety and others .The issue of the stability of these networks remains one of the work priorities for the development of these networks. According to the algorithms reviewed, the hybrid algorithm can be relied upon this process. There will be no interruption in communications, a short delay time in sending and receiving packets, and long-life time of cluster members.

The behavior of vehicles on the roads is of great importance in the classification process. By observing this behavior, it is possible to reach a data transfer control model that guarantees its arrival at the lowest possible time and cost.

Comparing the simulation results that were presented in the previous chapter, we find that the quality of the clustering is directly proportional to the range of the cluster. Therefore, to obtain a better performance of the proposed algorithm, the cluster range should be chosen as the largest range that is close to the data transmission range allowed by Wi-Fi connections. As for mobility speed, it is inversely proportional to the criteria for evaluating the quality of the proposed algorithm, so to obtain the desired results from this algorithm, the speed must be small as possible. Cluster formation time can be simply reducing by decrease the number of simulation nodes.

There are cases and scenarios in which a number of vehicles that may be the cluster header leave the road, thus interrupting communication between the vehicles and the base station.

Given the presence of technologies for obtaining location and direction, such as GPS and the Internet, which are equipped with most network vehicles, we proposed the improvement, given that there is no additional cost, as all the technologies required for this improvement are already present in the vehicles.

In the future, there are so many enhancements we can do to evolve the hybrid algorithm performance, and it is as follows:

- Linking between the clusters through get ways to facilitate the movement of vehicles between the clusters.

- Adding improvements to routing protocols, such as "AODV" to ensure that packets are delivered in an almost real time, and therefore higher accuracy in the algorithm results and shorter cluster formation time.

References

[1] X. Cheng and B. Huang, 'A Center-Based Secure and Stable Clustering Algorithm for VANETs on Highways', Wireless Communications and Mobile Computing, vol. 2019, pp. 1–10, 01 2019.

[2] M. Talib, A. Hassan, Z. Abas, A. Mohammed, and T. Abbas, 'A Center-Based Stable Evolving Clustering Algorithm with Grid Partitioning and Extended Mobility Features for VANETs', 09 2020.

[3] H.-W. Tseng, R.-Y. Wu, and C.-W. Lo, 'A stable clustering algorithm using the traffic regularity of buses in urban VANET scenarios', Wireless Networks, vol. 26, 05 2020.

[4] M. Talib, A. Hassan, B. Hussin, Z. Abas, Z. Talib, and Z. Sabah, 'A Novel Stable Clustering Approach based on Gaussian Distribution and Relative Velocity in VANETs', International Journal of Advanced Computer Science and Applications, vol. 9, 05 2018.

[5] G. Liu, N. Qi, J. Chen, C. Dong, and Z. Huang, 'Enhancing clustering stability in VANET: A spectral clustering-based approach', China Communications, vol. 17, pp. 140–151, 2020.

[6] G. Rossi, Z. Fan, W. H. Chin, and K. Leung, 'Stable Clustering for Ad-hoc Vehicle Networking', 03 2017, pp. 1–6.

[7] 'IEEE Standard for Information Technology - Telecommunications and information exchange between systems - Local and Metropolitan networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Higher Speed Physical Layer (PHY) Extension in the 2.4 GHz band', IEEE Std 802. 11b-1999, pp. 1–96, 2000.

[8] 'Evolution of 802.11 (physical layer)'. [Online]. Available: https://www.okob.net/texts/mydocuments/80211physlayer/.

[9] X. Cheng and B. Huang, 'A Center-Based Secure and Stable Clustering Algorithm for VANETs on Highways', Wireless Communications and Mobile Computing, vol. 2019, pp. 1–10, 01 2019.

[10] G. Rossi, Z. Fan, W. H. Chin, and K. Leung, 'Stable Clustering for Ad-hoc Vehicle Networking', 03 2017, pp. 1–6.

[11] 'SUMO at a Glance - SUMO Documentation'. [Online]. Available: https://sumo.dlr.de/docs/SUMO_at_a_Glance.html.

[12] E, 'Simulation of Urban MObility', 29-Jun-2023. [Online]. Available: http://sourceforge.net/apps/mediawiki/sumo/index.php?title=Main_Page.

[13] 'ns-3Tutorial---Tutorial'.[Online].Available: https://www.nsnam.org/docs/tutorial/html/index.html.

[14] E, 'Simulation of Urban MObility', 29-Jun-2023. [Online]. Available: http://sourceforge.net/apps/mediawiki/sumo/index.php?title=Tools/TraceExporte.

[15] O. Baradia and L. Jalalaltamimi, "Stable Clustering Algorithms for VANETs: A Survey," 2023 International Conference on Information Technology (ICIT), Amman, Jordan, 2023, pp. 637-642, doi: 10.1109/ICIT58056.2023.10226088.

[16] D. Kakan and Rayamajhi, Anjan & C. Mashrur. 'Vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication in a heterogeneous wireless network –

Performance evaluation ‘, Transportation Research Part C: Emerging Technologies. 68. 168-184. 10.1016/j.trc.2016.03.008.

[17] J. Y. Yu and P. H. J. Chong, "A survey of clustering schemes for mobile Ad-hoc networks," in *IEEE Communications Surveys & Tutorials*, vol. 7, no. 1, pp. 32-48, First Qtr. 2005, doi: 10.1109/COMST.2005.1423333.

[18] Singh, A., Kumar, M., Rishi, R., Madan, D.K. "A Relative Study of MANET and VANET: Its Applications, Broadcasting Approaches and Challenging Issues." In: Meghanathan, N., Kaushik, B.K., Nagamalai, D. (eds) Advances in Networks and Communications. CCSIT 2011. Communications in Computer and Information Science, vol 132. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-17878-8_63

Appendix

These attachments were mentioned throughout this thesis. They were arranged from 1 to 5 based on what was mentioned during the thesis.

Appendix 1.A

```
void DoSomething (double x, double y)
{
std::cout << Now().GetSeconds() << " : x=" << x << " y=" << y << std::endl;
}
int main (int args, char* argv[])
{
Simulator::Schedule (Seconds (3), &DoSomething, x, y);
...
}
```

Appendix 1.B

```
void PrintTheTime ()
{
std::cout << "Simulation time is: " << Now().GetSeconds() << std::endl;
//Schedule the next call after 1 second from now.
Simulator::Schedule (Seconds (1), &PrintTheTime);
}
...
int main (int args, char* argv[])
{
...
//schedule the first call after 1 second from the start of the simulation
Simulator::Schedule (Seconds (1), &PrintTheTime);
...
}
```

Appendix 2.A

```
Ptr<Node> some_node = CreateObject <Node> ();
```

Appendix 2.B

```
NodeContainer nodes;
nodes.Create (4); //Creates 4 nodes
Ptr<Node> n0 = nodes.Get (0); //pointer to node 0
```

Appendix 2.C

```
//Create a zero-payload packet
Ptr <Packet> p1 = Create <Packet> ();
//Create a packet with a payload of size 100 bytes (all zeros)
Ptr <Packet> p2 = Create <Packet> (100);
```

Appendix 3.A

```
//NodeList is a global list of all simulation nodes.
Ptr<Node> node = NodeList::GetNode (0);
for (uint32_t i=0; i<node->GetNDevices(); i++)
{
NetDevice dev = node->GetDevice (i);
if ( dev->GetInstanceTypeId() == WifiNetDevice::GetTypeId() )
{
Ptr<WifiNetDevice> wifi_dev = DynamicCast <WifiNetDevice> (dev);

}
else if (dev->GetInstanceTypeId() == CsmaNetDevice::GetTypeId())
{
Ptr<CsmaNetDevice> wifi_dev = DynamicCast <CsmaNetDevice> (dev);
//Maybe do something with it.
}
}
```

Appendix 3.B

```
ApplicationContainer apps = //some code;
Ptr<UdpEchoServer> server0 = DynamicCast <UdpEchoServer> (apps.Get(0));
```

Appendix 4.A

```
Ptr <ClassOne> obj_one = CreateObject <ClassOne> ();
Ptr <ClassTwo> obj_two = CreateObject <ClassTwo> ();
obj_one->AggregateObject (obj_two); //Now the two objects are associated
//...
Ptr <ClassTwo> s1 = obj_one->GetObject <ClassTwo> (); //s1 points to the same address as obj_two
Ptr <ClassOne> s2 = obj_two->GetObject <ClassOne> (); //s1 points to the same address as obj_one
//The following will cause a runtime error because you cannot aggregate two objects of the same type
Ptr <ClassTwo> tmp_two = CreateObject <ClassTwo> ();
obj_one->AggregateObject (tmp_two); // error!
```

Appendix 4.B

```
#include "ns3/mobility-module.h"
NodeContainer nodes;
nodes.Create (4);
MobilityHelper mob;
mob.SetMobilityModel ("ns3::ConstantPositionMobilityModel"); //stationary nodes
mob.Install (nodes); // Perform aggregation between every Node and ConstantPositionMobilityModel
//So far, all nodes are positioned at 0,0,0. Let's change the position of node 0
Ptr <Node> n0 = nodes.Get (0);
//the following works because MobilityModel is a subclass of ConstantPositionMobilityModel
Ptr <MobilityModel> mob_n0 = n0->GetObject <MobilityModel> ();
Vector previous_pos = mob_n0->GetPosition ();
//Set the position to 10,15,0. The unit is meters
mob_n0->SetPosition ( Vector (10,15,0));
std::cout << "Node was at " << previous_pos << " now in " << mob_n0->GetPosition () << std::endl;
```

Appendix 5.A

<algorithm name>_main.cc: Which contains the call of the main function. During the execution of the program, it is designed to accept at least four arguments with a certain order:
a)traceFile : the name of the file that contains the simulation details and has been produced with the TraceExporter (the SUMO's extension that was described previously).
b)nodeNum :number of nodes that participate in the simulation.
c)duration : the period of times that the simulation lasts.
logFile : the name of the simulation log file.

Appendix 5.B

•node-movements.h and node-movements.cc: These two files implement the class "NodeMovement" which represents the structure of details read from the input simulation file that is declared by the user. Each node keeps its nodeID, its position (int x, int y), its behavior, and its velocity (which is the total velocity on the x and y axes).
•custom-mobility-helper.h and custom-mobility-helper.cc: These two files implement the class "CustomMobilityHelper" which parse the input simulation file that is declared by the user and assigns extracted information to its related nodes.
v2v-control-client-helper.h and v2v-control-client-helper.cc: These two files implement the class "V2VControlClientHelper" which installs algorithms application script on the network nodes and pass parameters from the main program