

**NEW MULTI-HOP CLUSTERING ALGORITHM FOR
VEHICULAR AD HOC NETWORKS**

PROJECT REPORT

Submitted

in partial fulfillment of the requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY

in the faculty of

COMPUTER SCIENCE AND ENGINEERING

by

K. JAYARAM LAKSHMANA RAO [R.NO 16021A0527]

K. PRANAY VARMA [R.NO 16021A0530]

A. KEERTHI [R.NO 16021A0511]

Under the esteemed Guidance of

Dr. K. SAHADEVAIAH



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
UNIVERSITY COLLEGE OF ENGINEERING KAKINADA(A)
JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY
KAKINADA-533003, A.P, INDIA**

2019-2020

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
UNIVERSITY COLLEGE OF ENGINEERING KAKINADA(A)
JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY
KAKINADA-533003, A.P, INDIA



DECLARATION FROM THE STUDENTS

We hereby declare that the work described in this thesis, entitled *“New Multi-Hop Clustering Algorithm For Vehicular Ad Hoc Networks”*, which is being submitted by our team in partial fulfillment of the requirements for the award of the degree of *Bachelor of Technology* with specialization of Computer Science and Engineering to the Department of Computer Science and Engineering, University College of Engineering Kakinada(A), Jawaharlal Nehru Technological University Kakinada, is the result of investigation carried out by us under the supervision of Dr. K. Sahadevaiah, Professor, Department of Computer Science and Engineering, University College of Engineering Kakinada (A), Jawaharlal Nehru Technological University Kakinada - 533003.

The work is original and has not been submitted to any other University or Institute for the award of any degree or diploma.

Place : Kakinada

Signature:

Date :

K. J. LAKSHMANA RAO

[16021A0527]

K. PRANAY VARMA

[16021A0530]

A. KEERTHI

[16021A0511]

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
UNIVERSITY COLLEGE OF ENGINEERING KAKINADA (A)
JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY
KAKINADA – 533003, A.P., INDIA



CERTIFICATE FROM THE SUPERVISOR

This is to certify that the project work entitled “*New Multi-Hop Clustering Algorithm For Vehicular Ad Hoc Network*” that is being submitted by K. Jayaram Lakshmana Rao, K. Pranay Varma & A.Keerthi bearing registration numbers 16021A0527, 16021A0530 & 16021A0511 respectively in partial fulfilment of the requirements for the award of degree of *Bachelor of Technology* in the faculty of Computer Science & Engineering to the University College of Engineering (Autonomous), Jawaharlal Nehru Technological University Kakinada-533003, A.P. is a record of bonafide project work carried out by them under my guidance and supervision.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Signature of the Supervisor

Dr. K. Sahadevaiah

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
UNIVERSITY COLLEGE OF ENGINEERING KAKINADA (A)
JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY
KAKINADA – 533003, A.P., INDIA



CERTIFICATE FROM THE HEAD OF DEPARTMENT

This is to certify that the project work entitled ***“New Multi-Hop Clustering Algorithm For Vehicular Ad-Hoc Network”*** that is being submitted by K. Jayaram Lakshmana Rao, K. Pranay Varma & A. Keerthi bearing registration numbers 16021A0527, 16021A0530 & 16021A0511 respectively in partial fulfillment of the requirements for the award of degree of Bachelor of Technology(B.tech) in the faculty of Computer Science & Engineering to the University College of Engineering (Autonomous), Jawaharlal Nehru Technological University Kakinada-533003, A.P. is a record of bonafide project work carried out by them at our department.

Signature of Head of the Department

Dr. D. Haritha

ACKNOWLEDGEMENTS

We wish to thank our supervisor Dr. K. Sahadevaiah, Professor, Department of CSE for accepting and supporting us as project students. We sincerely convey our gratefulness and heartfelt to honorable and esteemed project guide for his supervision, guidance, encouragement, and counsel throughout the project. Without his invaluable advice and assistance, it would not have been possible for us to complete this thesis. His words of encouragement have often inspired me and improved our hopes for completing the project work.

We would like to thank Dr. D. Haritha, Professor, and Head of Computer Science and Engineering Department, UCEK, JNTUK for her fabulous support and useful remarks throughout the project.

We are thankful to Dr. B. Balakrishna, Principal, University College of Engineering Kakinada (Autonomous), JNTUK for his support and enlightenment during the project.

We thank all the teaching and non-teaching staff of the Department of Computer Science and Engineering for their support throughout our project work.

K. Jayaram Lakshmana Rao	- 16021A0527
K. Pranay Varma	- 16021A0530
A. Keerthi	- 16021A0511

ABSTRACT

As a hierarchical network architecture, the cluster architecture can improve the routing performance greatly for vehicular ad hoc networks (VANETs) by grouping the vehicle nodes. However, the existing clustering algorithms only consider the mobility of a vehicle when selecting the cluster head. The rapid mobility of vehicles makes the link between nodes less reliable in clusters. A slight change in the speed of cluster head nodes has a great influence on the cluster members and even causes the cluster head to switch frequently. These problems make the traditional clustering algorithms perform poorly in the stability and reliability of the VANET. A new multi-hop clustering algorithm is proposed to solve these problems. The algorithm is based on the idea of a multi hop clustering algorithm that ensures the coverage and stability of the cluster. In the cluster head selection phase, a priority-based neighbor-following strategy is proposed to select the optimal neighbor nodes to join the same cluster. This strategy makes the inter cluster nodes have high reliability and stability. By ensuring the stability of the cluster members and selecting the most stable node as the cluster head in the N-hop range, the stability of the clustering is greatly improved. In the cluster maintenance phase, by introducing the cluster merging mechanism, the reliability and robustness of the cluster are further improved. In order to validate the performance of the new algorithm, we do many detailed comparison experiments with the algorithms of single hop and N-HOP in the Network Simulator 2 environment.

CONTENTS

Acknowledgements	iv
Abstract	v
Contents	vi
List of Figures	ix
List of Abbreviations	x
Chapter 1 Introduction	1-7
1.1 An Overview of Vehicular Ad Hoc Networks	1
1.1.1 Characteristics of VANETs	3
1.1.2 Application of VANETs	4
1.2 Motivation of the Work	5
1.3 Objectives of the Thesis	6
1.4 Organization of the Thesis	6
Chapter 2 Survey of Literature	8-16
2.1 What is Clustering?	8
2.2 Need for Clustering	9
2.3 Types Of Clustering	9
2.4 Clustering Protocols	11
2.5 Cluster Formation and Maintenance Algorithms	11
2.5.1 Cluster Formation Algorithm	11
2.5.2 Cluster Maintenance Algorithms	12
2.6 Clustering algorithms in VANETs	12
Chapter 3 Proposed Algorithm	17-19
3.1 Introduction	17

3.2 Information Table	18
3.3 Priority Neighbour Following Strategy	18
3.4 Procedure	19
3.5 Software and Hardware Requirements	19
3.5.1 Software Requirments	19
3.5.2 Hardware Requirements	19
Chapter 4 Design of Proposed Algorithm	20-26
4.1 System Model	20
4.2 Flowchart	21
4.3 Design Diagrams	22
Chapter 5 Simulative Study of Proposed Algorithm	27-59
5.1 Introduction to Network Simulator-2	27
5.1.1 Installation of the Network Simulator-2	27
5.1.2 Salient Features of Network Simulator-2	28
5.1.3 Network Simulator-2 Architecture	29
5.1.4 Network Simulator-2 Components	29
5.1.5 Creating Wireless Topology	31
5.1.6 Simulation Environment	31
5.2 Structure of Program Execution	32
5.3 Modules	32
5.3.1 Dynamic Creation of Nodes	32
5.3.2 Information Table Generation	39
5.3.3 Cluster Formation	42
5.3.4 Cluster Head Selection	43

5.3.5 Cluster Merging	44
Chapter 6 Experimental Results and Analysis	60-65
6.1 Output Format	60
6.2 Performance Analysis	61
6.2.1 Time Vs Cluster Head Changes	61
6.2.2 Time Vs Average Cluster Head Duration	62
6.2.3 Time Vs Cluster Member Changes	63
6.2.4 Time Vs Average Cluster Member Duration	64
Chapter 7 Conclusion and Scope of Future Work	66
7.1 Conclusion	66
7.2 Future Work	66
Chapter 8 References	67-71

LIST OF FIGURES

Fig No	Figure Name	Page No
2.1	Vehicular Ad Hoc Network	9
2.2	Types of Clustering	10
2.3	Static Clustering	10
4.1	Architecture Model	16
4.2	Algorithm Flow chart	17
4.3	Use-case Diagram	20
4.4	Statechart Diagram	21
4.5	Activity Diagram	22
5.1	Block Diagram of Architecture of Network Simulator-2	25
5.2	NAM Screen	26
5.3	Simulation Parameters	28
5.4	Execution Structure	28
5.5	Output of Module-1	35
5.6	Output of Module-2	38
5.7	Output of Module-3	39
5.7	Output of Module-4	40
5.9	Output of Module-5 Before Merging	46
5.10	Output of Module-5 After Merging	46
6.1	Output Format	57
6.2	Time vs Cluster Head Changes	59
6.3	Time vs Average Cluster Head Duration	60
6.4	Time vs Cluster Member Changes	61
6.5	Time vs Average Cluster Member Duration	62

LIST OF ABBREVIATIONS

Abbreviation		Expansion
VANETs	-	Vehicular Ad Hoc Networks
MANETs	-	Mobile Ad Hoc Networks
IVC	-	Inter Vehicular Communication
RSU	-	Road Side Units
CH	-	Cluster Head
CM	-	Cluster Member
CG	-	Cluster Gateway
V2V	-	Vehicle-To-Vehicle
V2I	-	Vehicle-To-Infrastructure
CBLR	-	Cluster Based Location Routing
UML	-	Unified Modeling Language
NS2	-	Network Simulator-2
VINT	-	Virtual Inter Network TestBed
NAM	-	Network Animator
SUMO	-	Simulation of Urban Mobility
MCA-V2I	-	Multi-hop Clustering Approach Vehicle-to-Internet
VMaSC	-	Vehicular Multi-hop Algorithm for Stable Clustering

Chapter 1

Introduction

1.1 An Overview of Vehicular Ad Hoc Networks

Vehicular Ad Hoc Networks (VANETs) The networks that interconnect vehicles on road are called Vehicular Ad hoc Networks (VANETs). “A Mobile Ad hoc Network (MANET) consists of mobile nodes that connect themselves in a decentralized, self-organizing manner and may also establish multi-hop routes. If mobile nodes are cars, this is called a vehicular ad hoc network”. Several different applications are emerging in VANETs. These applications include safety applications to make driving much safer, mobile commerce and other information services that will inform drivers about any type of congestion, driving hazards, accidents, traffic jams. VANETs have several different aspects compared to MANETs, in that the nodes move with high velocity because of which the topology changes rapidly. VANETs are also prone to several different attacks. Therefore, the security of VANETs is indispensable. VANETs pose many challenges on technology, protocols, and security, which increase the need for research in this field.

A Vehicular Ad Hoc Network or VANET is a technology that uses moving cars as nodes in a network to create a mobile network. VANET turns every participating car into a wireless router or node, allowing cars approximately 100 to 300 meters from each other to connect and, in turn, create a network with a wide range. As cars fall out of the signal range and drop out of the network, other cars can join in, connecting vehicles to one another so that a mobile network is created. It is estimated that the first

systems that will integrate this technology are police and fire vehicles to communicate with each other for safety purposes.

VANETs come under the category of *Wireless Ad hoc Networks*. In vehicular ad-hoc networks, the node may be a *vehicle* or the *road side units*. They can communicate with each other by allowing the wireless connection up to a particular range.

Inter-Vehicular Communications (IVC) also known as vehicular ad hoc networks (VANETs) have become very popular in recent years. A Vehicular Adhoc Network is a special type of Mobile Ad hoc Networks (MANETs is a kind of wireless ad hoc network and is a self-configuring network of mobile routers connected by wireless links) which use vehicles as nodes. The main difference is that mobile routers which build the network are vehicles like cars or trucks.

Several different applications are emerging with regard to vehicular communications. For example, safety applications for safer driving, information services to inform drivers about the driving hazards and other business services in the vicinity of the vehicle. Government, corporations, and the academic communities are working on enabling new applications for VANETs. A main goal of VANETs is to increase road safety by the use of wireless communications. To achieve these goals, vehicles act as sensors and inform each other about abnormal and potentially hazardous conditions like accidents, traffic jams and glazes. Vehicular networks closely resemble ad hoc networks because of their rapidly changing topology. Therefore; VANETs require secure routing protocols. Numerous Applications are unique to the vehicular setting. These applications include safety applications that will make driver safe, mobile commerce, roadside services that can intelligently inform drivers about congestion, businesses, and services in the vicinity of the Vehicle. VANETs, especially compared to MANETs are

characterized by several unique aspects. Nodes move with high velocity, resulting in high rates of topology changes. Because of rapidly changing topology due to vehicle motion, the vehicular network closely resembles an ad hoc network. The constraints and optimizations are remarkably different. From the network perspective, security and scalability are two significant challenges. A formidable set of abuses and attacks become possible. Hence, the security of vehicular networks is indispensable. The growing importance of inter vehicular communications (IVC) has been recognized by the government, corporations, and the academic community.

1.1.1 Characteristics of VANETs

The main characteristics of VANETs are :

Highly dynamic topology: The high speed of the vehicles along with the availability of choices of multiple paths defines the dynamic topology of VANETs.

Frequent disconnected network: The high speed of the vehicles in one way defines the dynamic topology whereas on the other hand necessitates the frequent requirements of the roadside unit lack of which results in frequent disconnections.

Mobility modeling and Prediction: The prediction of vehicle position and their movements is very difficult. This features of mobility modeling and prediction in VANETs is based on the availability of predefined roadmaps models. The speed of the vehicles is again important for efficient network design.

Communication Environment: Once we are having a mobility model, yet we are not done. As the mobility model may have different features depending upon road architecture, highways, or city environments. Communicating in these situations has to be taken care of.

Hard delay constraints: At the time of emergency, delivery of messages on time is a critical problem. Therefore, handling such situations rather than talking only about high data rates is not sufficient.

Interaction with onboard sensors: Sensors are the mode of communications. Sensors can read data related to velocity of the vehicle, direction and can communicate to the data center. Thus sensors can be used in link formation and in routing protocols.

Unlimited Battery Power and Storage: Nodes in VANETs do not suffer power and storage limitations as in sensor networks; therefore optimizing duty cycle is not as relevant as in sensor networks.

1.1.2 Application of VANETs

The RSU can be treated as an access point or router or even a buffer point which can store data and provide data when needed . All data on the RSUs are uploaded or downloaded by vehicles.

Real-time traffic: The real time traffic data can be stored at the RSU and can be available to the vehicles whenever and wherever needed. This can play an important role in solving the problems such as traffic jams, avoid congestions and in emergency alerts such as accidents etc.

Post Crash Notification: A vehicle involved in an accident would broadcast warning messages about its position to trailing vehicles so that it can take decision with time in hand as well as to the highway patrol for tow away support

Traffic Vigilance: The cameras can be installed at the RSU that can work as input and act as the latest tool in low or zero tolerance campaign against driving offenses

Digital map downloading: Map of regions can be downloaded by the drivers as per the requirement before traveling to a new area for travel guidance. Also, Content Map Database Download acts as a portal for getting valuable information from mobile hotspots or home stations.

Route Diversions: Route and trip planning can be made in case of road congestions.

Parking Availability: Notifications regarding the availability of parking in the metropolitan cities helps to find the availability of slots in parking lots in a certain geographical area.

Active Prediction: It anticipates the upcoming topography of the road, which is expected to optimize fuel usage by adjusting the cruising speed before starting a descent or an ascent. Secondly, the driver is also assisted.

Cooperative Collision Warning: Alerts two drivers potentially under crash route so that they can mend their ways .

Remote Vehicle Personalization Diagnostics: It helps in downloading of personalized vehicle settings or uploading of vehicle diagnostics from infrastructure.

1.2 Motivation of the Work

The importance of vehicular ad hoc networks become more popular these days due to its *less power requirement, low cost, performance and high potential application areas*. They have been deployed for a wide variety of applications, including real time traffic, post crash notification, traffic vigilance, parking availability and many more.

Clustering is a technique that *groups vehicles nearby*. They elect a cluster head so that they communicate with other cluster heads and Road Side Units(RSU) to make inter cluster communication possible thus reducing the total burden on the RSU. Because of clustering only CH communicates with RSU rather than communicating each and every vehicle communicating with RSU. By making an efficient clustering we can decrease the cluster changes and cluster head switches.

1.3 Objectives of the Thesis

The main objectives of the thesis are:

- To obtain a stable cluster formation.
- Consider speed and direction while clustering.
- Obtain Information Table
- To elect a stable cluster head.
- Cluster merging must take place when cluster interference occurs.

1.4 Organization of the Thesis

The thesis has eight chapters:

Chapter 1: Gives *introduction*. In this chapter a brief overview of the project has been given that project motivation and objective. A brief overview of Characteristics of VANETs and Application of VANETs is also given.

Chapter 2: Presents the *survey of literature*. In this chapter we discuss the project domain and the detailed description of existing systems by analyzing the literature survey of the existing techniques.

Chapter 3: Presents a *proposed algorithm*. This chapter gives an overview of the algorithm that is proposed

Chapter 4: Discusses the *design of proposed algorithm*. This chapter mainly consists of design and unified modeling language diagrams: use case diagrams, state chart diagrams and sequence diagrams.

Chapter 5: Discusses the *simulative study of proposed algorithm*. This chapter introduces network simulator-2 and modules of the accurate algorithm.

Chapter 6: Gives *experimental results and analysis*. This chapter consists of snapshots of results and analysis.

Chapter 7: Records *conclusion and scope of future work*. This chapter mainly consists of conclusion for the project and future enhancement.

Chapter 8: *Provides references.* This chapter consists of all the references made to implement this algorithm.

Chapter 2

Survey of Literature

2.1 What is Clustering

Clustering is a mechanism of grouping of vehicles based upon some predefined metrics such as density, velocity, and geographical locations of the vehicles. Clustering in Vehicular Ad hoc Networks (VANET) is one of the control mechanisms for dynamic topology.

VANET nodes are characterized by their *high mobility*, and the existence of VANET nodes in the same geographic proximity does not mean that they exhibit the same mobility patterns. Therefore the clustering schemes of VANET should consider the speed and velocity of nodes to construct a stable clustering structure.

Each cluster has *cluster members* (CM), *cluster head* (CH) and *cluster Gateway*(CG). Inside the cluster one node that coordinates the cluster activities is cluster head (CH). Inside the cluster, there are ordinary nodes also that have direct access only to this one cluster head, and cluster head forward information through gateway nodes. Gateways are nodes that can hear two or more cluster heads that will be different clusters.

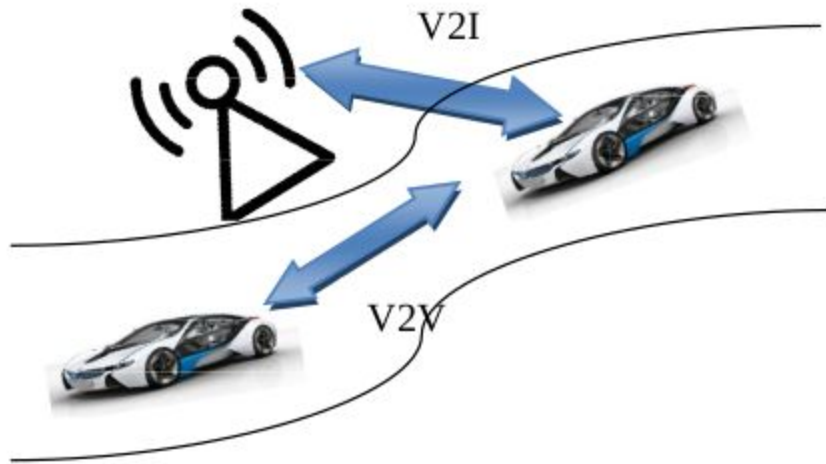


Figure 2.1: Vehicular Ad Hoc Network

2.2 Need for Clustering

We need the clustering for following reasons:

- Efficient Communication V2V & V2I
- Decrease Routing load
- Increase Packet delivery
- Bandwidth fully utilization

2.3 Types of Clustering:

Clustering is divided into two subcategories according to the nature of cluster formation.

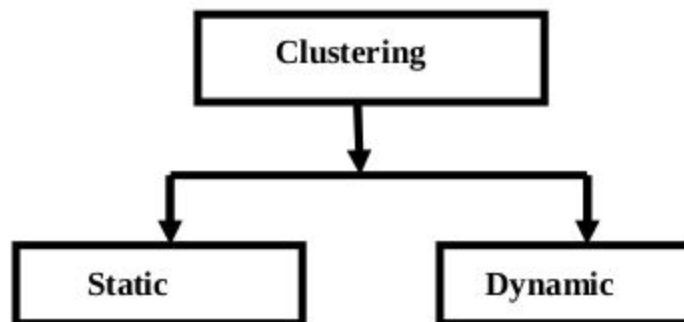


Figure 2.2: Types of Clustering

Static clustering: In this type, a stable cluster is formed. Sometimes these clusters also contain RSU. In this case the cluster works within the range of the RSU. Static clusters move in the same direction with the same speed. There is no need for reconfiguration of clusters in static clustering. These clusters are not scalable. Cluster formation and maintenance is easy for static clustering. Routing protocols are easily designed. But scalability and other factors decrease the performance of this network.

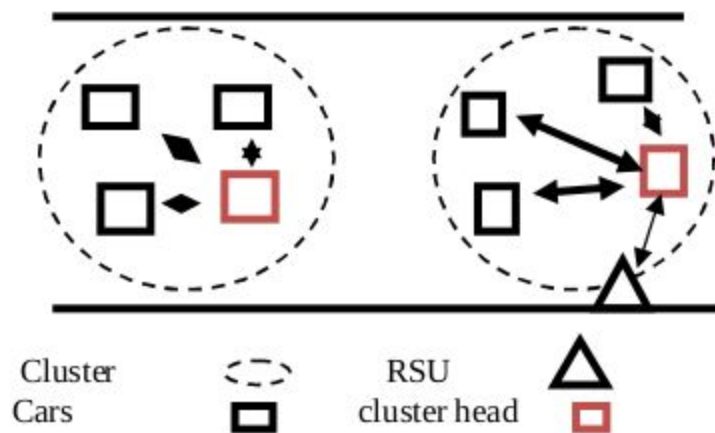


Figure 2.3: Static Clustering

Dynamic clustering: In this type, cluster formation is done dynamically in minimum time. Due to the dynamic nature of the network cluster reconfiguration is needed. Cluster heads are changed because of high mobility. Cluster reconfiguration and range of cluster head depends on the density of the area. These clusters are easily scalable.

2.4 Clustering Protocols

Clustering protocols partitioned the network into clusters on the bases of speed; direction etc. clustering based routing protocols are –

Mobility Based protocols: Formation of cluster and selection of cluster head is based on the mobility factor .other parameters such as speed and direction are also used. This category further divided into two sub categories –

1. Lane based
2. Speed based

Non Mobility Based protocols: These protocols do not consider mobility factors. It depends on the *density of the network*. Clusters are formed automatically and communication depends on the gateway nodes. One of the non mobility protocols is cluster based location routing (CBLR).

Certificate Based Protocols: These protocols are used for *privacy preservation*. Certificate generation and revocation is done in this protocol.

2.5 Cluster Formation and Maintenance Algorithms

Clustering algorithms are designed to make cluster process *efficient* and *secure*. Mainly the following types of clustering algorithms are developed.

2.5.1 Cluster Formation Algorithm:

A cluster is a small group of vehicles containing a cluster head, a gateway node and more than one member. Formation algorithm is developed to make clusters, cluster head selection, choose gateway and enable communication.

2.5.2 Cluster Maintenance Algorithms:

These algorithms are used to recover the links and cluster from any type of failure. A member node is dead when the cluster does not receive a message sent by that node. A node rejoins the cluster when it stops receiving the messages sent by the cluster head. Maintenance algorithms also describe the following methods-

1. joining the cluster
2. leaving the cluster
3. merging the cluster
4. resigning procedure of cluster head.

Single hop :Single hop clustering is that each node in the network is not more than 1-hop away from a cluster head.

Multi hop :Multi hop clustering is the one in which each node can have multiple hops from the cluster head.

Double headed :In this double headed clustering each cluster can contain one or two cluster heads in it, the primary CH and secondary SCH.

Distributed :A D-hop clustering algorithm, called DHCV is the one which organizes vehicles into non overlapping clusters which have adaptive sizes according to their respective mobility. The D-hop clustering algorithm creates clusters in such a way that each vehicle is at most D hops away from a cluster head.

2.6 Clustering Algorithms in VANETs

Samo Vodopivec and Janez Bester (2012) [9] proposed that clustering is a technique for grouping nodes in geographical vicinity together, making the network more robust and scalable. This article presents an overview of proposed clustering algorithms for use in vehicular ad-hoc networks (VANET). We survey different clustering algorithms and highlight their objectives, features, specialties and

possible limitations. Varieties of different approaches have been observed whereby typically each one focuses on different performance metrics. Diverse are also complexities of algorithms and the input data they use and rely on.

Oussama Senouci, Zibounda Aliouat and Saad Harous (2019)

[10] proposed a new Multi-hop Clustering Approach over Vehicle-to-Internet called MCA-V2I to improve VANETs' performance. MCA-V2I is based on the reasonable assumption that a vehicle can connect to the Internet via a special infrastructure called a Road Side Unit Gateway. Once connected to the Internet, each vehicle can obtain and share the necessary information about its Multi-hop neighbors to perform the clustering process. This latter is performed using a Breadth-first search (BFS) algorithm for traversing a graph based on a Mobility Rate that is calculated according to mobility metrics. MCA-V2I strengthens clusters' stability through the selection of a Slave Cluster Head in addition to the Master Cluster Head. We evaluate the performances of the proposed scheme using network simulator NS-2 and the VanetMobiSim integrated environment.

Ahmed Alioua and Sidi-Mohammed Senouci [14] have proposed a work that aims to facilitate the management of the disconnected infrastructure-less VANET areas by organizing the network topology using a distributed multi-hop clustering algorithm. The proposed algorithm is an enhanced version of the distributed version of LTE for V2X communications (LTE4V2X-D) [7] framework for the infrastructure-less VANET zone. We are able to improve the performance of LTE4V2X-D to better support clustering stability while decreasing clustering overhead. This is made possible due to a judicious choice of metrics for the selection of cluster heads and maintenance of clusters. Our algorithm uses a combination of three metrics, vehicle direction,

velocity and position, in order to select a cluster-head that will have the longest lifetime in the cluster. The simulation comparison results of the proposed algorithm with LTE4V2X-D demonstrate the effectiveness of the novel enhanced clustering algorithm through the considerable improvement in the cluster stability and overhead.

Jyotsna rao Dawande and Sanjay Silakari (2015) [15] have proposed a new protocol "enhanced distributed multi-hop clustering algorithm for VANETs based on neighborhood follow (EDMCNF) collaborated with Road Side Unit" to overcome these problems. Here, the new vehicles will directly communicate with RSU to take information about the stable cluster for it which will reduce the hello packet overhead and also reduces the time taken for cluster selection. The reduction of hello packet overheads and reduction of time taken for cluster selection jointly results in the reduction of average overheads.

Zhenxia Zhang and Azzedine Boukerche (2011) [13] have proposed a new mobility metric is introduced to represent relative mobility between vehicles in multi-hop distance. Extensive simulation experiments are run using ns2 to demonstrate the performance of our clustering scheme. To test the clustering scheme under different scenarios, both the Manhattan mobility model and the freeway mobility model are used to generate the movement paths for vehicles.

Sanaz Khakpour (2013) [12] proposed a clustering mechanism based on a novel VANET compatible algorithm, which will be evaluated and tested in the context of vehicular target tracking. The key features of this algorithm comprise higher cluster head and cluster lifetime and distributed cluster head selection mechanism.

Soufiane Ouahou, Slimane Bah and Zohra Bakkoury (2017) [11] have proposed a novel multi-hop clustering model including: cluster construction/destruction mechanism, multi-hop links establishment and

cluster head election algorithm based on a new metric. Indeed, the proposed metric captures real environment parameters. The simulation results show that our clustering solution performs better than existing solutions especially when the obstacle shadowing model is used which is the most likely scenario in real situations.

Seyhan Ucar, Sinem Coleri Ergen (2013) [16] have proposed VMaSC: Vehicular Multi-hop algorithm for Stable Clustering. VMaSC is a novel clustering technique based on choosing the node with the least mobility calculated as a function of the speed difference between neighboring nodes as the cluster heads through multiple hops. Extensive simulation experiments performed using ns-3 with the vehicle mobility input from the Simulation of Urban Mobility (SUMO) demonstrate that novel metric used in the evaluation of the least mobile node and multi-hop clustering increases cluster head duration by 25% while decreasing the number of cluster head changes by 10%.

Rajan Jamgekar (2018) [5] introduced a new architecture based on IEEE 802.11p for multi hop clustering algorithm. It achieves high data packet ration and low delay while keeping the usage of the cellular architecture at minimum level.

Mengying Ren, Lyes Khoukhi and Jun Zhang (2016) [24] have proposed a new mobility-based and stability-based clustering algorithm (MSCA) for urban city scenario, which makes use of the vehicle's moving direction, relative position and link lifetime estimation. We evaluate the performance of our proposed algorithm in terms of changing maximum lane speed and traffic flow rate. Our proposed algorithm performs well in terms of average cluster head lifetime and average number of clusters.

Meysam Azizian, Soumaya Cherkaoui, Abdelhakim Senhaji Hafid (2016) [6] presented a distributed clustering algorithm, called

DCEV, which constructs multi-hop clusters. DCEV places vehicles into non-overlapping clusters which have adaptive size based on their relative mobility. The cluster formation is based on a D-hop clustering scheme where each node selects its cluster head at most D-hop distance. To create clusters, DCEV uses a new metric to let vehicles choose the most stable route to their desired cluster head within their D-hop neighbourhood. For this purpose, each node calculates the mean relative mobility value of each discovered route (end-to-end relative mobility). DCEV considers the route which has the least end-to-end relative mobility as the most stable route. Extensive simulations were conducted for different scenarios to validate the performance of DCEV clustering algorithm. Results show that DCEV efficiently manages to build stable clusters.

Chapter 3

Proposed Algorithm

3.1 Introduction:

As a hierarchical network architecture, the cluster architecture can improve the routing performance greatly for Vehicular Ad hoc Networks (VANETs) by grouping the vehicle nodes. However, the existing clustering algorithms only consider the mobility of a vehicle when selecting the cluster head. The rapid mobility of vehicles makes the link between nodes less reliable in clusters. A slight change in the speed of cluster head nodes has a great influence on the cluster members and even causes the cluster head to switch frequently. These problems make the traditional clustering algorithms perform poorly in the stability and reliability of the VANET.

Our New Multi Hop clustering algorithm is based on the idea of a multi-hop clustering algorithm that ensures the coverage and stability of the cluster. In the cluster head selection phase, a priority-based neighbor-following strategy is proposed to select the optimal neighbor nodes to join the same cluster. This strategy makes the inter-cluster nodes have high reliability and stability. By ensuring the stability of the cluster members and selecting the most stable node as the cluster head in the N-hop range, the stability of the clustering is greatly improved. In the cluster maintenance phase, by introducing the cluster merging mechanism, the reliability and robustness of the cluster are further improved.

3.2 Information Table:

We assume that each vehicle is equipped with an on-board unit with a maximum communication radius R and communicates with other vehicles by the WAVE communication protocol. An information table (INFO_TABLE) is stored in each vehicle, which contains motion-related information of vehicle nodes within a predefined maximum hop count (MAX_HOP). The INFO-TABLE mainly includes the vehicle's unique ID, vehicle direction, speed and location-related information. When the vehicle state is changed or receives a HELLO packet from neighbor node, the INFO_TABLE table is modified. The HELLO packet is broadcast to its neighbor node. Its main information includes node ID, speed, direction, CH_ID and hop count to CH node. If a node does not receive a HELLO packet from a neighbor node during a specified time interval, the route entry is deleted when the time-stamp expires.

3.3 Priority Neighbour Following Strategy :

In the traditional multi-hop clustering algorithm, vehicle nodes and cluster head nodes with less relative distance are added to the cluster as cluster members to form a stable cluster structure. However, in the multi-hop cluster architecture, it is difficult for a vehicle node to get precise motion information from its multi-hop nodes. A priority neighbor following mechanism is proposed. In the cluster formation phase, a vehicle is not required to proactively detect the cluster head nodes within multi-hop distances. However, it is necessary to select the most stable node within its one hop range by the priority neighbor following strategy and share the same cluster head. They are merged into the same cluster. Next, we introduce the priority neighbor following strategy in detail and give the calculation method.

No_followers= direct_f +indirect_f

- No_followers : Total no of followers of a vehicle.
- direct_f : followers in one hop.
- indirect_f : followers of the connected vehicles.

The more the no of stable followers the more priority to become the cluster head.

3.4 Procedure

- Create dynamic vanet nodes.
- Generate an info table of every node.
- priority calculation of every node.
- Formation of cluster.
- Election of cluster head.
- Cluster merging if required.

3.5 Software and Hardware Requirements

3.5.1 Software Requirements

- Operating system : Ubuntu
- Languages : NS2 , Python3, C++
- Python version : 3.x
- NS2 version : 2.35

3.5.2 Hardware Requirements

- System : Pentium dual core
- Hard disk :120 GB
- Monitor : 15" LED
- Input Devices: Keyboard, Mouse
- Ram : 1GB

Chapter 4

Design Of Proposed Algorithm

4.1 System Model

The hybrid network architecture fully combines the advantages of a lower network overhead and flexible deployment of V2V and the advantages of lower transmission latency and a wide propagation range of LTE.

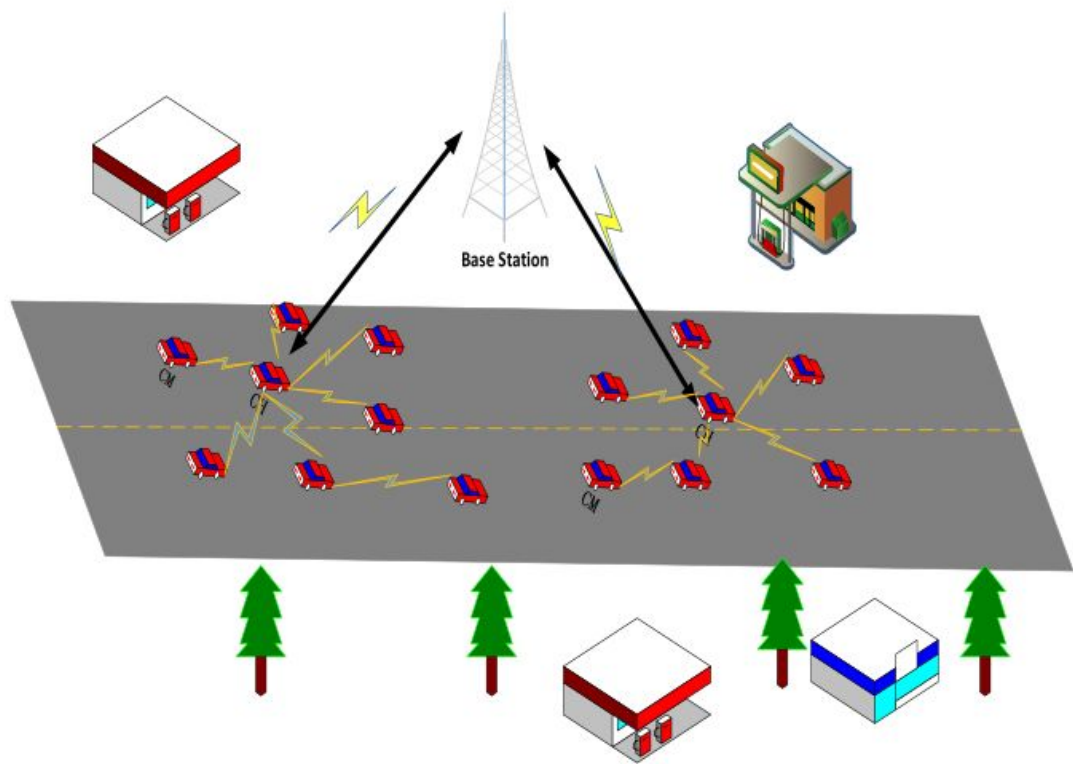


Figure 4.1:Architecture Model

4.2 Flowchart

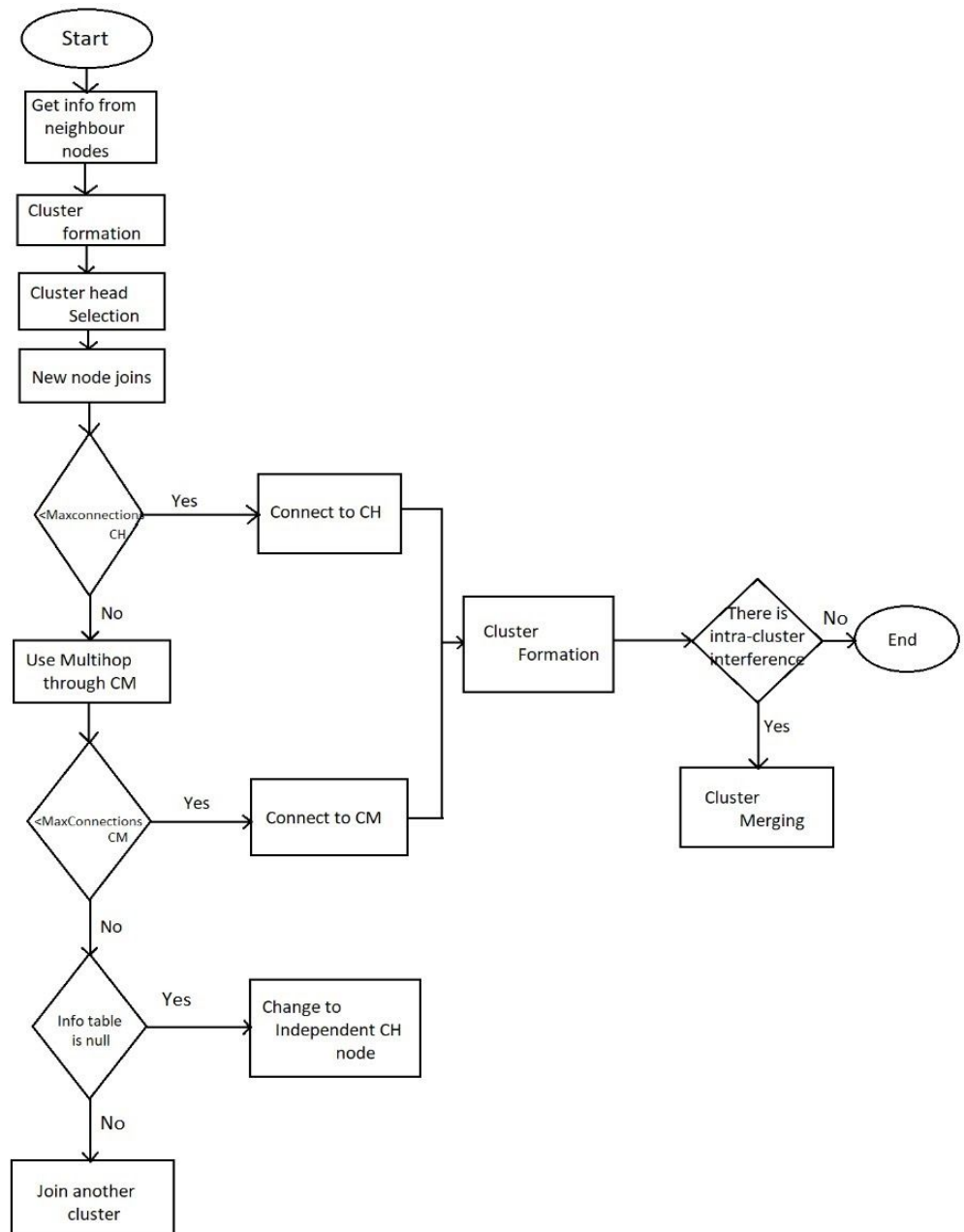


Figure 4.2:Algorithm Flow chart

4.3 Design Diagrams

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML consists of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software systems, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

Goals:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extensibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming language and development process.
4. Provide a formal basis for understanding the modeling language.

5. Encourage the growth of the OO tools market.

A UML system is represented using five different views that describe the system from a distinctly different perspective. Each view is defined by a set of diagrams, which is as follows.

- User Model View
- Structural model View
- Behavioral Model View
- Implementation Model View
- Environmental Model View

Use-case Diagram: A use-case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

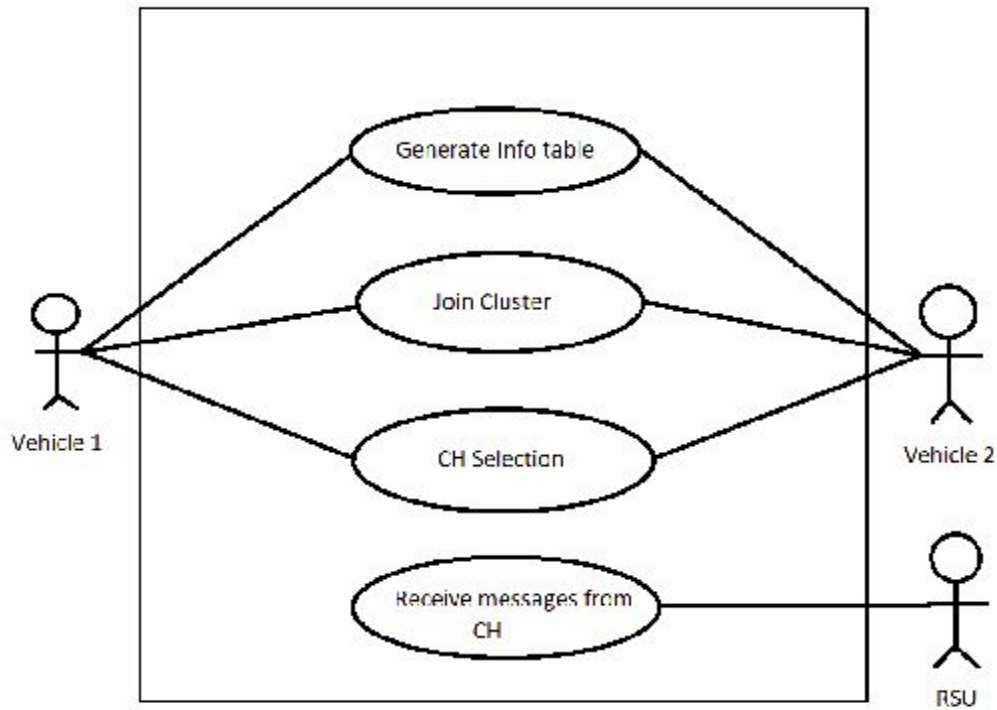


Figure 4.3: Use-case Diagram

Statechart Diagram: A Statechart diagram describes a state machine. State machine can be defined as a machine which defines different states of an object and these states are controlled by external or internal events.

Statechart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of Statechart diagram is to model the lifetime of an object from creation to termination.

Following are the main purposes of using Statechart diagrams –

- To model the dynamic aspect of a system.
- To model the life time of a reactive system.
- To describe different states of an object during its lifetime.

- Define a state machine to model the states of an object.

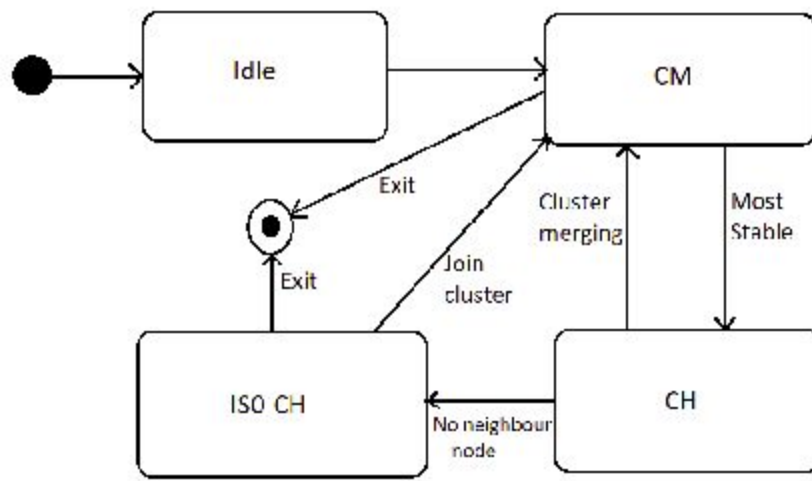


Figure 4.4: Statechart Diagram

Activity Diagram: It is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The basic purpose of activity diagrams is to capture the dynamic behavior of the system.. It is also called object-oriented flowchart.

This UML diagram focuses on the execution and flow of the behavior of a system instead of implementation. Activity diagrams consist of activities that are made up of actions that apply to behavioral modeling technology.

Activity Diagram Notations :

- *Initial State:* The starting state before an activity takes place is depicted using the initial state.
- *Action or Activity State:* An activity represents execution of an action on objects or by objects. We represent an activity using a rectangle with rounded corners. Basically any action or event that takes place is represented using an activity.

- *Action Flow or Control flows:* Action flows or Control flows are also referred to as paths and edges. They are used to show the transition from one activity state to another.
- *Final State or End State:* The state which the system reaches when a particular process or activity ends is known as a Final State or End State. We use a filled circle within a circle notation to represent the final state in a state machine diagram. A system or a process can have multiple final states.

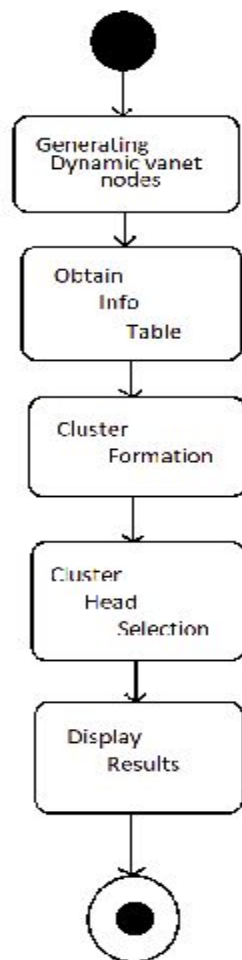


Figure 4.5: Activity Diagram

Chapter 5

Simulative Study Of Proposed Algorithm

5.1 Introduction to Network Simulator-2

Network Simulator (NS2) is a discrete event driven simulator developed at UC Berkeley. It is part of the VINT project.

The goal of NS2 is to support networking research and education. It is suitable for designing new protocols, comparing different protocols and traffic evaluations. NS2 is developed as a collaborative environment. It is distributed freely and open source. A large number of institutes and people in development and research use, maintain and develop NS2. Versions are available for FreeBSD, Linux, Solaris, Windows.

5.1.1 Installation of the Network Simulator-2

The installation steps for network simulator ns-allinone-2.35 are as follows:

- (1) Check for the complete LINUX installation (with all the options enabled during installation) in the system.
- (2) During installation of NS2 it's better to get into the ROOT login.
- (3) Copy the given ns-allinone-2.35.tar.gz file in the root directory.
- (4) Extract the files in the same folder using extract option or the command in the terminal as
root>tar zxvf ns-allinone-2.35.tar.gz
- (5) The files will be extracted in the newly created folder as nsallinone-2.35.
- (6) Now go to the terminal and get into the folder ns-allinone-2.35 and do the following
Root> cd ns-allinone-2.35
Root/ns-allinone-2.35 >./install

(7) This command will run for 10-15 minutes. If it is a successful installation you will be getting IMPORTANT NOTICE notes (detailing about the paths to set)

(8) Copy the ns-21 file given into the following folder

Computer/file system/etc/profile.d

(9) Now close all the terminals and open a new terminal and get into the folder root/ns-allinone-2.35 and give ns command and you will be getting as below when you each time press ENTER key which indicates that the installation is complete .

```
Root/ns-allinone-2.35>ns
```

```
%
```

```
%exit
```

(10) Now you can run the programs in a new terminal

5.1.2 Salient Features of Network Simulator-2

Abstraction: Detailed and high-level simulations possible in NS2. The detail of an individual protocol to the aggregation of multiple data flows and the interaction of multiple protocols.

Emulation: Emulation allows a running simulator to interact with operational network nodes.

Scenario generation: Automatic creation of complex traffic patterns, topologies, and dynamic events (link failures) can help generate such scenarios.

Extensibility: The simulator must be easy to extend if its users are to add new functionality, explore a range of scenarios, and study new protocols. NS makes it easy.

Visualization: Visualization using the network animation tool nam provides a dynamic representation that allows researchers to develop better.

5.1.3 Network Simulator-2 Architecture

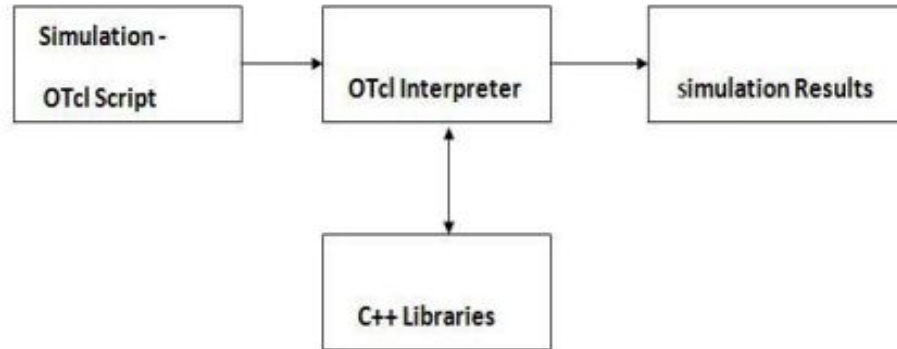


Figure 5.1: Block diagram of architecture of Network Simulator-2

The architecture of NS2 contains the following:

- OTcl: Object-oriented support

- tclcl: C++ and Otcl linkage

- Discrete Event Scheduler

- Data network (the Internet) components

5.1.4 Network Simulator-2 Components

Network simulator has different components to study the internal procedure and performance of the protocols which are implemented in the simulator. The following are the list of components.

- NAM: Nam is a Tcl/TK based animation tool for viewing network simulation traces and real world packet traces. It supports topology layout, packet level animation, and various data inspection tools

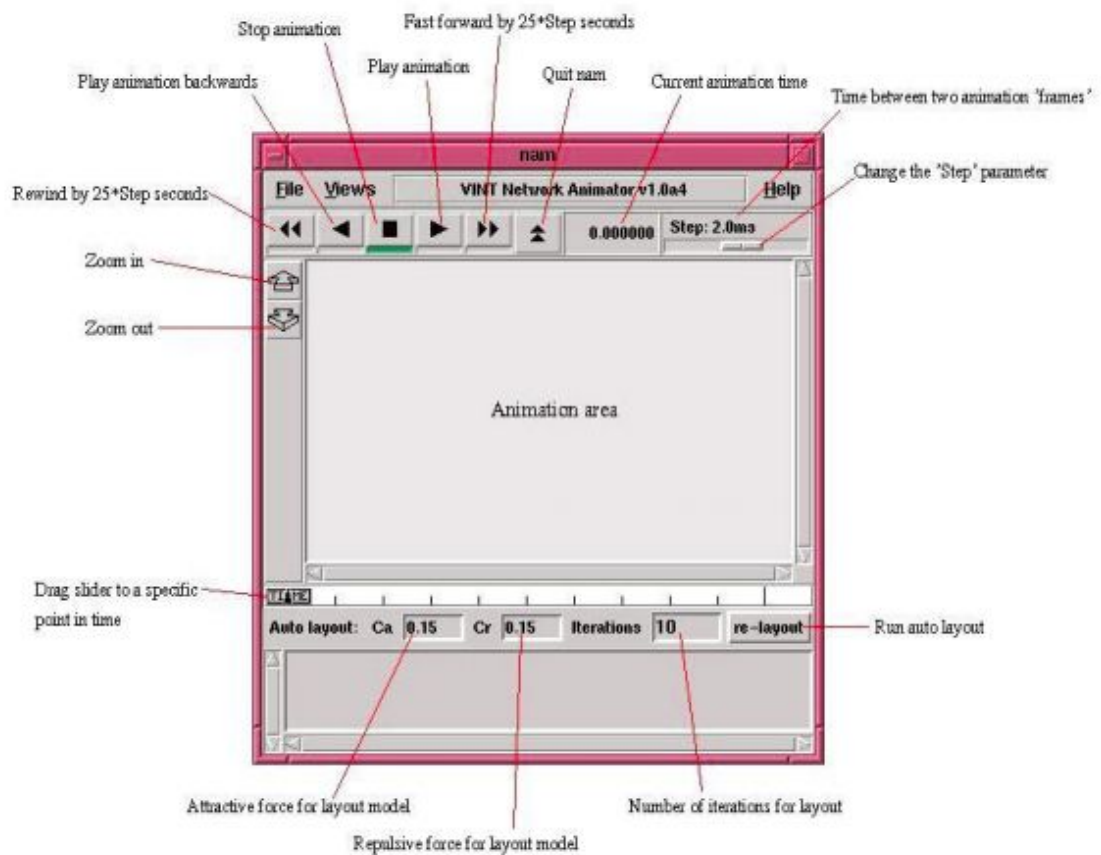


Figure 5.2: NAM Screen

The nam began at LBL. It has evolved substantially over the past few years. The nam development effort was an ongoing collaboration with the VINT project. Currently, it is being developed as an open source project hosted at Sourceforge.

5.1.5 Creating Wireless Topology

Mobile Node is the basic nsNode object with added functionalities like movement, ability to transmit and receive on a channel that allows it to be used to create mobile, wireless simulation environments. The class Mobile Node is derived from the base class Node. Mobile Node is a split object. The mobility features including node movement, periodic position updates, maintaining topology boundary etc are implemented in C++ while plumbing of network components within mobile Node itself (like classifiers, dmux, LL, Mac, Channel etc) have been implemented.

5.1.6 Simulation Environment

Parameter name	Value
Mobility model	Random Way point model
Simulation area	1000m x 1000m
Routing protocol	AODV
MAC protocol	Way point model IEEE802.11
Maximum speed	10m/s+
Simulation time	20s
Number of nodes	Dynamic number of nodes

Figure 5.3: Simulation Parameters

5.2 Structure of Program Execution:

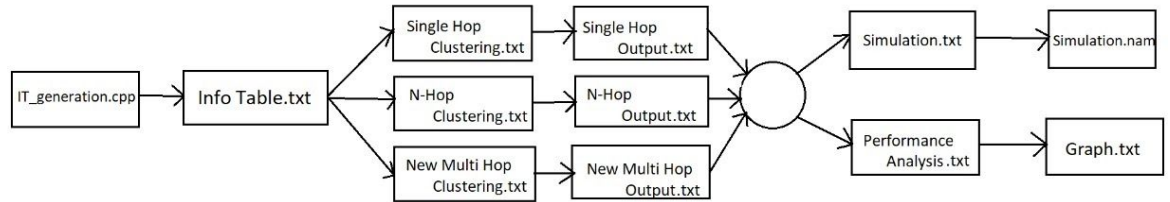


Figure 5.4: Execution Structure

5.3 Modules

The implementation of New Multi Hop Clustering algorithm is divided into five modules and is done on NS2 (Network Simulator 2) environment.

- 1) Creation of Dynamic vanet nodes
- 2) Information Table Generation
- 3) Cluster Formation
- 4) Cluster Head Selection
- 5) Cluster Merging
- 6) Performance Analysis

5.3.1 Dynamic Creation of Nodes

In the first module we generate the traffic dynamically. So that, on the obtained traffic we can implement our new multihop clustering algorithm.

CODE:

```
set val(chan)          Channel/WirelessChannel    ;# channel type
set val(prop)          Propagation/TwoRayGround    ;# radio-propagation
model
set val(netif)         Phy/WirelessPhyExt          ;# network interface type
set val(mac)           Mac/802_11Ext              ;# MAC type
set val(ifq)           Queue/DropTail/PriQueue     ;# interface queue type
set val(ll)            LL                          ;# link layer type
set val(ant)           Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)        50                          ;# max packet in ifq
set val(nn)            22                          ;# number of mobilenodes
set val(rp)            DSDV                       ;# routing protocol

set ns_                [new Simulator]
set tracefd            [open out.tr w]
set namtrace           [open out.nam w]
$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace 2000 2000
set topo              [new Topography]
$topo load_flatgrid 4000 4000
create-god $val(nn)
set col(0) yellow
set col(1) violet
set col(2) orange
set col(3) blue
set col(4) brown
set col(5) chocolate
set col(6) coral
set col(7) azure
```

```

set col(8) firebrick
set col(9) fuchsia
set col(10) chartreuse
set col(11) red
set col(12) green
# configure node
    $ns_ node-config -adhocRouting $val(rp) \
        -llType $val(ll) \
        -macType $val(mac) \
        -ifqType $val(ifq) \
        -ifqLen $val(ifqlen) \
        -antType $val(ant) \
        -propType $val(prop) \
        -phyType $val(netif) \
        -channelType $val(chan) \
        -topoInstance $topo \
        -agentTrace ON \
        -routerTrace ON \
        -macTrace OFF \
        -movementTrace OFF

set n(0) [$ns_ node]
$n(0) set X_ 180
$n(0) set Y_ 980
$n(0) set Z_ 0.0
$ns_ initial_node_pos $n(0) 40
$n(0) color "black"
set n(1) [$ns_ node]
$n(1) set X_ 110

```

```

$n(1) set Y_ 980
$n(1) set Z_ 0.0
$ns_ initial_node_pos $n(1) 40
$n(1) color "black"
set n(2) [$ns_ node]
$n(2) set X_ 40
$n(2) set Y_ 980
$n(2) set Z_ 0.0
$ns_ initial_node_pos $n(2) 40
$n(2) color "black"
set n(3) [$ns_ node]
$n(3) set X_ 180
$n(3) set Y_ 630
$n(3) set Z_ 0.0
$ns_ initial_node_pos $n(3) 40
$n(3) color "black"
set n(4) [$ns_ node]
$n(4) set X_ 110
$n(4) set Y_ 630
$n(4) set Z_ 0.0
$ns_ initial_node_pos $n(4) 40
$n(4) color "black"
$ns_ at 0.0 "$n(0) setdest 1000 825 60.0"
$ns_ at 0.0 "$n(1) setdest 1000 825 60.0"
$ns_ at 0.0 "$n(2) setdest 1000 825 60.0"
$ns_ at 0.0 "$n(3) setdest 750 675 70.0"
$ns_ at 0.0 "$n(4) setdest 750 675 70.0"
$ns_ at 8.3 "$n(3) setdest 1000 825 30.0"
$ns_ at 8.3 "$n(4) setdest 1000 825 30.0"

```

```

$ns_ at 14.0 "$n(0) setdest 1500 825 80.0"
$ns_ at 15.3 "$n(1) setdest 1500 825 80.0"
$ns_ at 16.6 "$n(2) setdest 1500 825 80.0"
$ns_ at 18.2 "$n(3) setdest 1500 825 30.0"
$ns_ at 19.5 "$n(4) setdest 1500 825 30.0"
$ns_ at 20.6 "$n(0) setdest 1700 900 70.0"
$ns_ at 21.7 "$n(1) setdest 2000 825 80.0"
$ns_ at 22.8 "$n(2) setdest 1700 825 60.0"
$ns_ at 35.0 "$n(3) setdest 1700 900 50.0"
$ns_ at 37.1 "$n(4) setdest 1800 825 50.0"
$ns_ at 23.6 "$n(0) setdest 1700 1500 70.0"
$ns_ at 21.7 "$n(1) setdest 2500 825 80.0"
$ns_ at 22.8 "$n(2) setdest 1700 675 60.0"
$ns_ at 34.3 "$n(1) setdest 3370 975 80.0"
$ns_ at 27.0 "$n(2) setdest 1800 40 60.0"
$ns_ at 39.5 "$n(3) setdest 1700 1500 40.0"
$ns_ at 43.4 "$n(4) setdest 2500 825 50.0"
$ns_ at 57.9 "$n(4) setdest 3370 975 50.0"
set c100 [$ns_ node]
$c100 set X_ 1000
$c100 set Y_ 600
$c100 set Z_ 0.0
$ns_ initial_node_pos $c100 0
$c100 color "black"
set c101 [$ns_ node]
$c101 set X_ 2500
$c101 set Y_ 600
$c101 set Z_ 0.0
$ns_ initial_node_pos $c101 0

```

```

$c101 color "black"
set c102 [$ns_ node]
$c102 set X_ 1500
$c102 set Y_ 1400
$c102 set Z_ 0.0
$ns_ initial_node_pos $c102 0
$c102 color "black"
$ns_ at 0.0 "$c150 label RSU1"
$ns_ at 0.0 "$c151 label RSU2"
$ns_ at 0.0 "$c152 label RSU3"
$ns_ at 0.0 "$c153 label RSU4"
$ns_ at 0.0 "$c150 add-mark m2 green hexagon"
$ns_ at 0.0 "$c151 add-mark m2 green hexagon"
$ns_ at 0.0 "$c152 add-mark m2 green hexagon"
$ns_ at 0.0 "$c153 add-mark m2 green hexagon"
$ns_ simplex-link $c112 $c104 100.0Mb 10ms DropTail
$ns_ simplex-link $c113 $c114 100.0Mb 10ms DropTail
$ns_ simplex-link $c115 $c114 100.0Mb 10ms DropTail
$ns_ simplex-link $c116 $c100 100.0Mb 10ms DropTail
set fp [open "NewMultihopPythonOutput.txt" r]
set file_data [read $fp]
set data [split $file_data "\n"]
foreach line $data {
    set data_1 [split $line "a"]
    set time [lindex $data_1 0]
    set ch_list_raw [lindex $data_1 1]
    set ch_list [split $ch_list_raw " "]
    set no_of_clusters [lindex $ch_list 0]
    set set_counter 2

```



```

        set only_ch_list [lreplace $ch_list 0 0]
        set    only_ch_list    [lreplace    $only_ch_list    $no_of_clusters
$no_of_clusters]
        if {$no_of_clusters!=""} {
            while {$set_counter<=$no_of_clusters+1} {
                set cm_list [lindex $data_1 $set_counter]
                foreach cm $cm_list {
                    set  ch_no  [lindex  $only_ch_list  [expr
$set_counter-2]]

                    $ns_ at $time "$n($cm) delete-mark m2"
                    $ns_ at $time "$n($cm) color $col($ch_no)"
                    if { $cm == $ch_no } {
                        $ns_ at $time "$n($cm) add-mark
m2 green hexagon"

                        }
                    }
                    incr set_counter 1
                }
            }
        }
    }
}

close $fp
$ns_ at 150.0 "stop"
$ns_ at 150.01 "#puts \"NS EXITING...\" ; $ns_ halt"
proc stop {} {
    global ns_ tracefd
    $ns_ flush-trace
    close $tracefd
    exec nam out.nam
    exit 0
}

```

```
}
$ns_run
```

Screenshot:

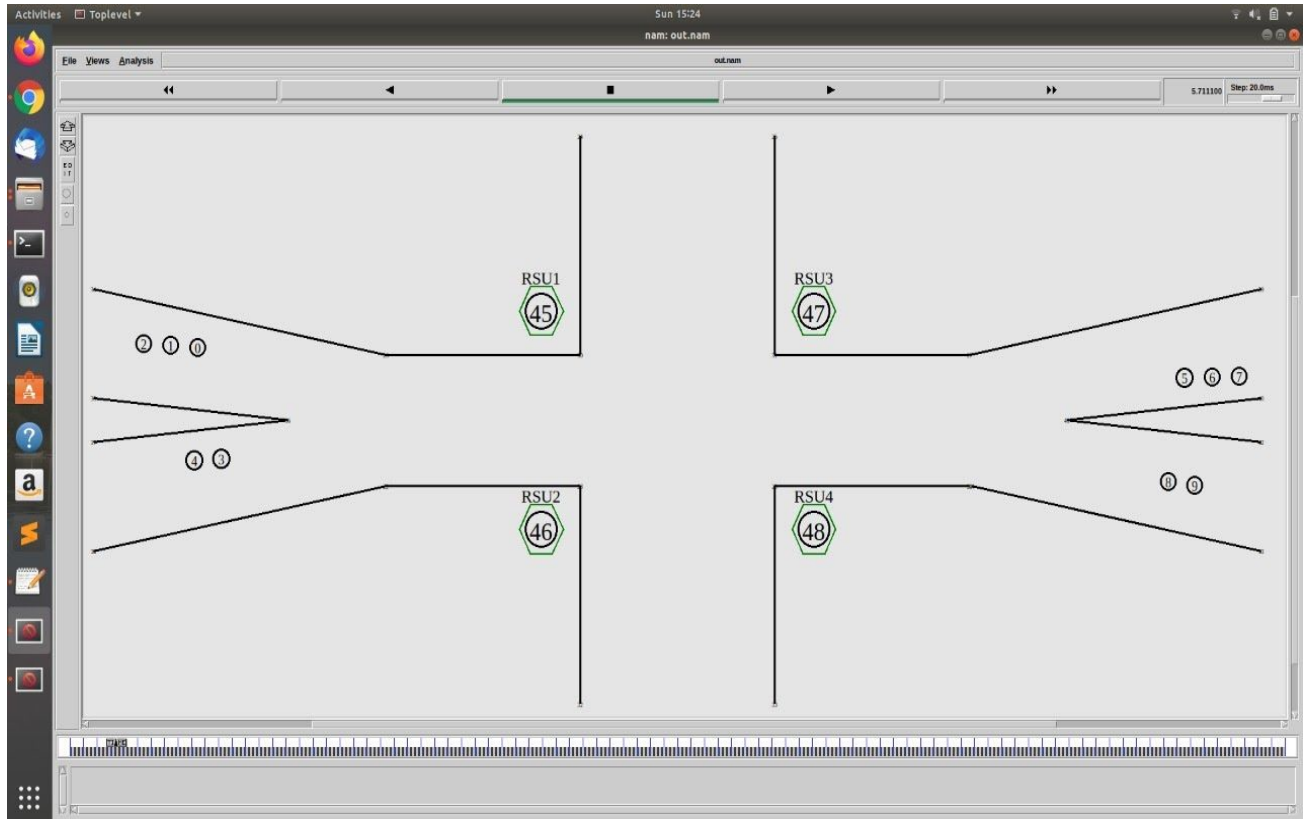


Figure 5.5: Output of Module-1

5.3.2 Information Table Generation

In this module the INFO_TABLE is generated. In VANET, a neighbor table called INFO_TABLE is maintained in each vehicle, which contains its own vehicle information and the information of neighbour nodes.

It is obtained by the traffic generated from the before module. It contains all the data (coordinates, speed, direction) about the vehicles which are in the range of a particular vehicle.

CODE:

```
#include<stdio.h>
#include<fstream>
using namespace std;
int main()
{
    double x[1200],y[1200],v[1200];
    int i;
    ofstream myfile ("InfoTable.txt");
    for(i=0;i<=1200;i++)
    {
        if(i<=140)
            x[i]=180.0+(820/14.0)*(i/10.0);
        else if(i>140 && i<=206)
            x[i]=1000.0+(500/6.6)*(i/10.0-14);
        else if(i>206 && i<=236)
            x[i]=1500.0+(200/3.0)*(i/10.0-20.6);
        else
            x[i]=1700;

        myfile << x[i] << " " ;
    }
    myfile<<"\n";

    for(i=0;i<=1200;i++)
    {
        if(i<=153)
            y[i]=980.0-(155/15.3)*(i/10.0);
```

```

        else if(i>153 && i<=343)
            y[i]=825.0;
        else if(i>343 && i<=450)
            y[i]=825.0+(150/10.7)*(i/10.0-34.3);
        else
            y[i]=975;

        myfile << y[i] << " " ;
    }
    myfile<<"\\n";

    for(i=0;i<=1200;i++)
    {
        if(i<=140)
            v[i]=60.0;
        else if(i>140 && i<=206)
            v[i]=80.0;
        else if(i>206 && i<=236)
            v[i]=70.0;
        else
            v[i]=70.0;

        myfile << v[i] << " " ;
    }
    myfile<<"\\n";
    return 0;
}

```

The image shows a screenshot of the Visual Studio Code editor interface. At the top, the title bar indicates 'Visual Studio Code' and the current file is 'example.txt'. The menu bar includes 'File', 'Edit', 'Selection', 'View', 'Go', 'Run', 'Terminal', and 'Help'. The sidebar on the left contains icons for Explorer, Search, Source Control, and Run and Debug. The main editor area displays a C++ file named 'example.txt' with line numbers 1 through 24. The code is a loop that prints numbers from 1 to 96, with some numbers repeated. The bottom status bar shows 'Ln 65, Col 4 (3 selected)' and 'UTF-8 LF Plain Text'. The terminal at the bottom shows the command 'bash' and the prompt 'jayaram@jayaram-HP-Notebook:~\$'.

5.3.3 Cluster Formation

By the obtained information table clustering algorithms (New, nhop, 1hop) are performed on it.

Screenshot:

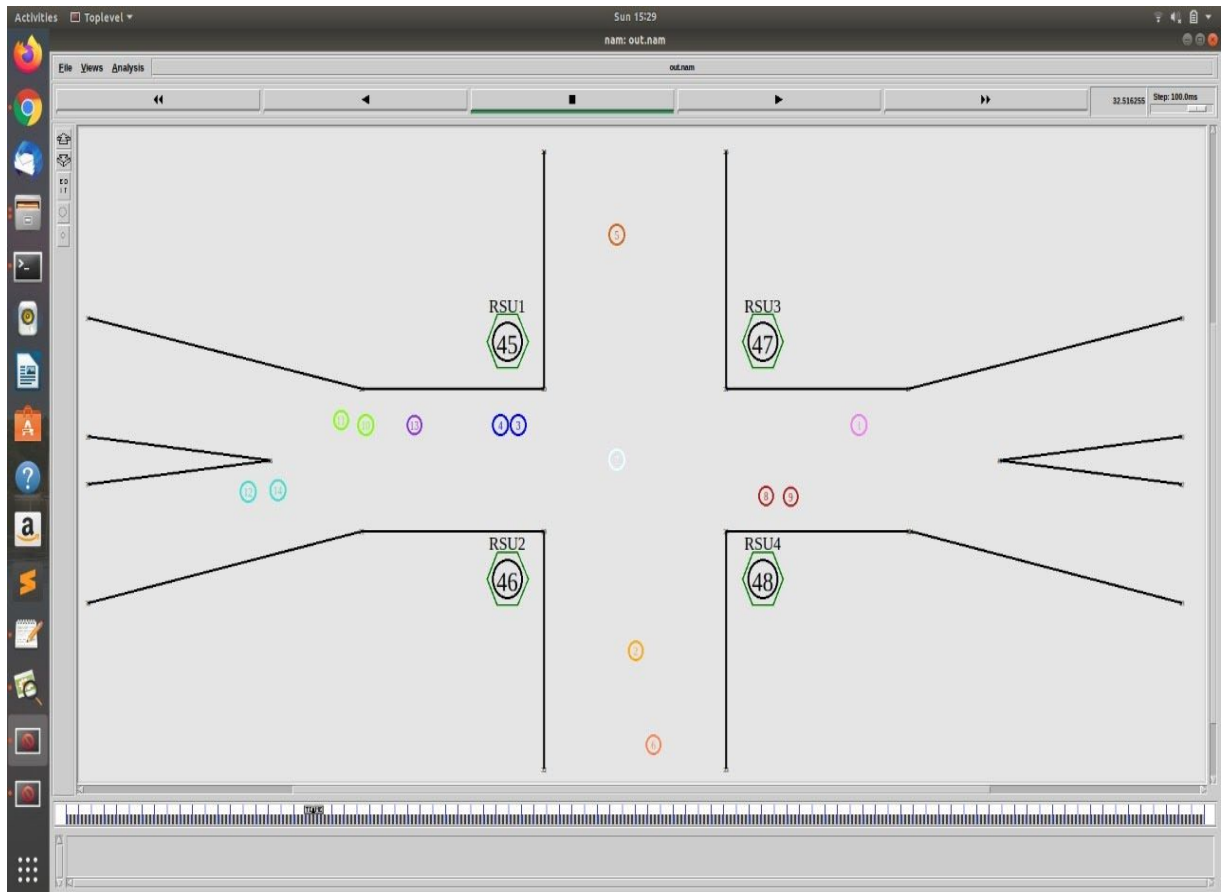


Figure 5.7: Output of Module-3

5.3.4 Cluster Head Selection

In VANET, it is important to select a stable cluster head because its topology changes rapidly, which can improve not only the stability of clustering, but also the lifetime of clusters. Selected the most stable cluster head to minimise the no. of cluster head changes

Focused on the velocity, direction and no. of neighbour nodes to select the most stable cluster head among the vehicles in the formed cluster.

Screenshot:

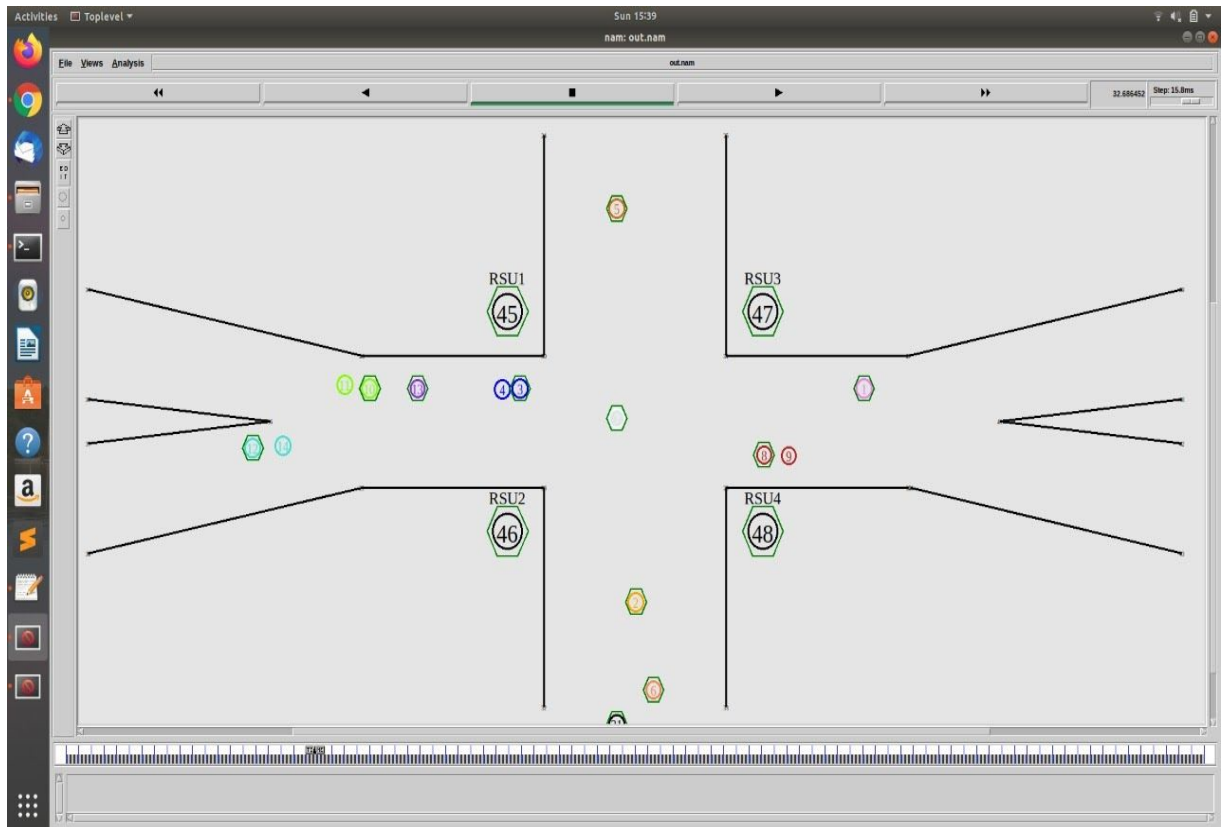


Figure 5.8: Output of Module-4

5.3.5 Cluster Merging

This module is about Cluster Merging. When the cluster head nodes in two clusters become neighbor nodes due to the change of vehicle speed, the clusters will overlap, causing inter-cluster interference.

During the merging process, the cluster head periodically detects whether the two neighbor clusters can be merged.

New Multi-hop Clustering .py

```
import math
neigh_dist = 100
neigh_vel = 60
class Graph:
    def __init__(self, V):
        self.V = V
        self.adj = [[] for i in range(V)]
    def DFSUtil(self, temp, v, visited):
        visited[v] = True
        temp.append(v)
        for i in self.adj[v]:
            if visited[i] == False:
                temp = self.DFSUtil(temp, i, visited)
        return temp
    def addEdge(self, v, w):
        self.adj[v].append(w)
        self.adj[w].append(v)
    def connectedComponents(self):
        visited = []
        cc = []
        for i in range(self.V):
            visited.append(False)
        for v in range(self.V):
            if visited[v] == False:
                temp = []
                cc.append(self.DFSUtil(temp, v, visited))
        return cc
    def angle_update_fun(s_u):
```



```

if (s_u != sim_units - 1):
    for i in range(no_of_vehicles):
        if (x[i][s_u + 1] - x[i][s_u] == 0):
            if (y[i][s_u + 1] - y[i][s_u] > 0):
                angle[i] = 90
            elif (y[i][s_u + 1] - y[i][s_u] < 0):
                angle[i] = 270
            elif (y[i][s_u + 1] - y[i][s_u] == 0):
                angle[i] = -1
        else:
            slope = (y[i][s_u + 1] - y[i][s_u]) / (x[i][s_u + 1] - x[i][s_u])
            angle[i] = math.degrees(math.atan(slope))
            if (y[i][s_u + 1] - y[i][s_u] < 0 and angle[i] > 0):
                angle[i] += 180
            if (y[i][s_u + 1] - y[i][s_u] < 0 and angle[i] < 0):
                angle[i] += 360
            if (y[i][s_u + 1] - y[i][s_u] > 0 and angle[i] < 0):
                angle[i] += 180
            if (x[i][s_u + 1] - x[i][s_u] < 0 and angle[i] == 0):
                print(x[i][s_u + 1] - x[i][s_u] < 0 and angle[i] == 0)
                angle[i] = 180

file1 = open("InfoTable.txt", "r")
f = open("NewMultihopPythonOutput.txt", "w")
total_file_list = file1.readlines()
no_of_lines = len(total_file_list)
x = []
y = []
v = []
no_of_vehicles = no_of_lines // 3

```

```

for i in range(no_of_lines):
    line = total_file_list[i]
    if (i < no_of_vehicles):
        x.append(list(map(float, line.split())))
        print(x)
    elif (i < 2 * (no_of_vehicles)):
        y.append(list(map(float, line.split())))
    else:
        v.append(list(map(float, line.split())))
sim_units = len(x[0])
print(v)
dist = [[0 for i in range(no_of_vehicles)] for j in range(no_of_vehicles)]
angle = [0] * no_of_vehicles
prev_ch_list=[]
for s_u in range(sim_units):
    g = Graph(no_of_vehicles)
    angle_update_fun(s_u)
    for i in range(no_of_vehicles):
        for j in range(i, no_of_vehicles):
            dist[i][j] = math.sqrt(math.pow(x[i][s_u] - x[j][s_u], 2) +
math.pow(y[i][s_u] - y[j][s_u], 2) * 1.0)
            dist[j][i] = math.sqrt(math.pow(x[i][s_u] - x[j][s_u], 2) +
math.pow(y[i][s_u] - y[j][s_u], 2) * 1.0)
            if (dist[i][j] < neigh_dist and abs(v[i][s_u]-v[j][s_u])<neigh_vel and
(abs(angle[i]-angle[j])<=50 or abs(angle[i]-angle[j])>=310 or angle[i]==-1 or
angle[j]==-1) ):
                g.addEdge(i, j)
                g.addEdge(j, i)
cc = g.connectedComponents()

```

```

def no_of_neighbours(node):
    counter = 0
    for dist_bet in dist[node]:
        if (dist_bet < neigh_dist):
            counter += 1
    return counter
ch_list = []
for cluster in cc:
    max_nei = -1
    for c_m in cluster:
        if (no_of_neighbours(c_m) > max_nei):
            max_nei = no_of_neighbours(c_m)
            ch = c_m
            if(ch not in prev_ch_list):
                for c_m_1 in cluster:
                    if(c_m_1 in cluster):
                        ch=c_m_1
                        break
    ch_list.append(ch)
prev_ch_list=ch_list
print(ch_list)
print(cc)
s_u_u = s_u / 10
f.write("%.1f" % s_u_u)
f.write("a")
f.write("%d " % len(ch_list))
for c_h in ch_list:
    f.write("%d " % c_h)
for cluster in cc:

```

```

f.write("a")
for c_m in cluster:
    f.write("%d " % c_m)
f.write("\n")
f.close()
file1.close()

```

Screenshot:
Before Merging

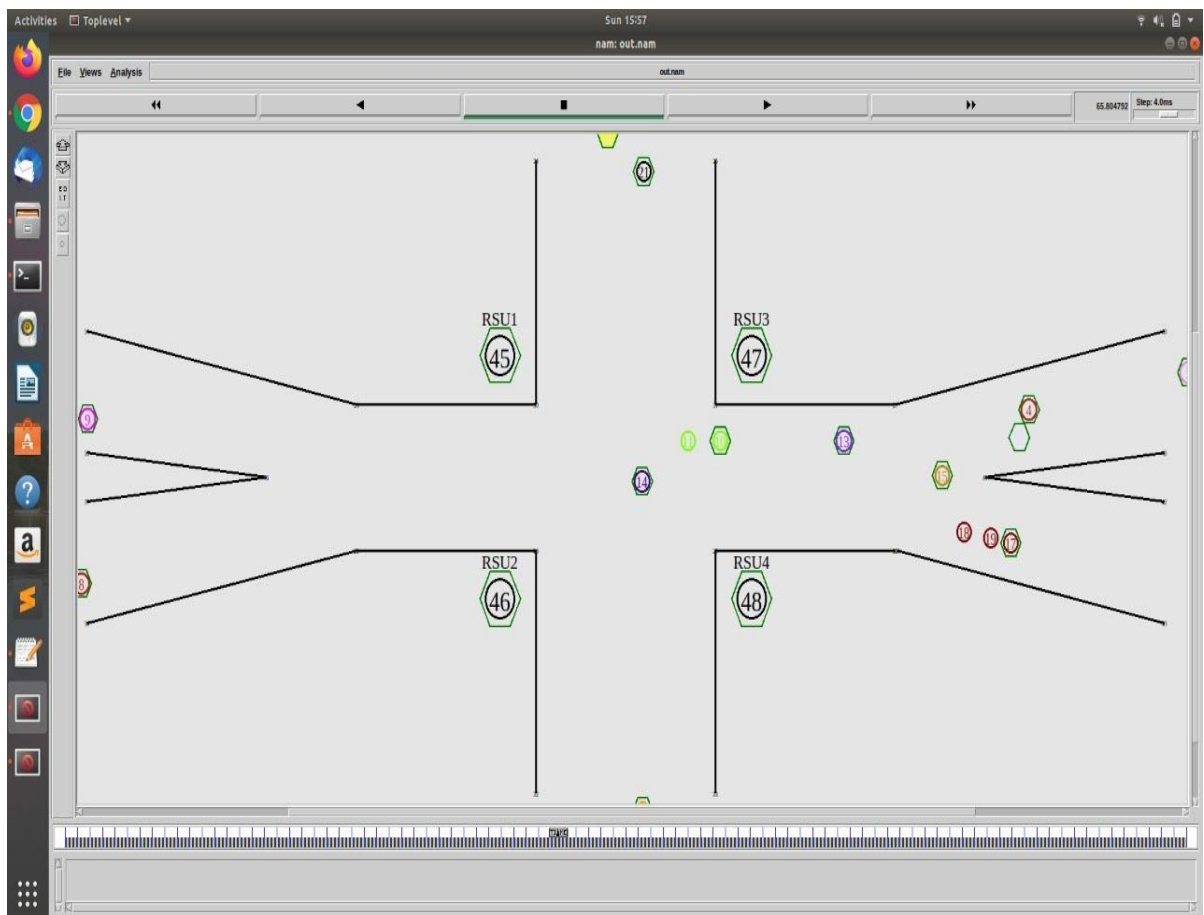


Figure 5.9: Output of Module-5 before merging

After Merging

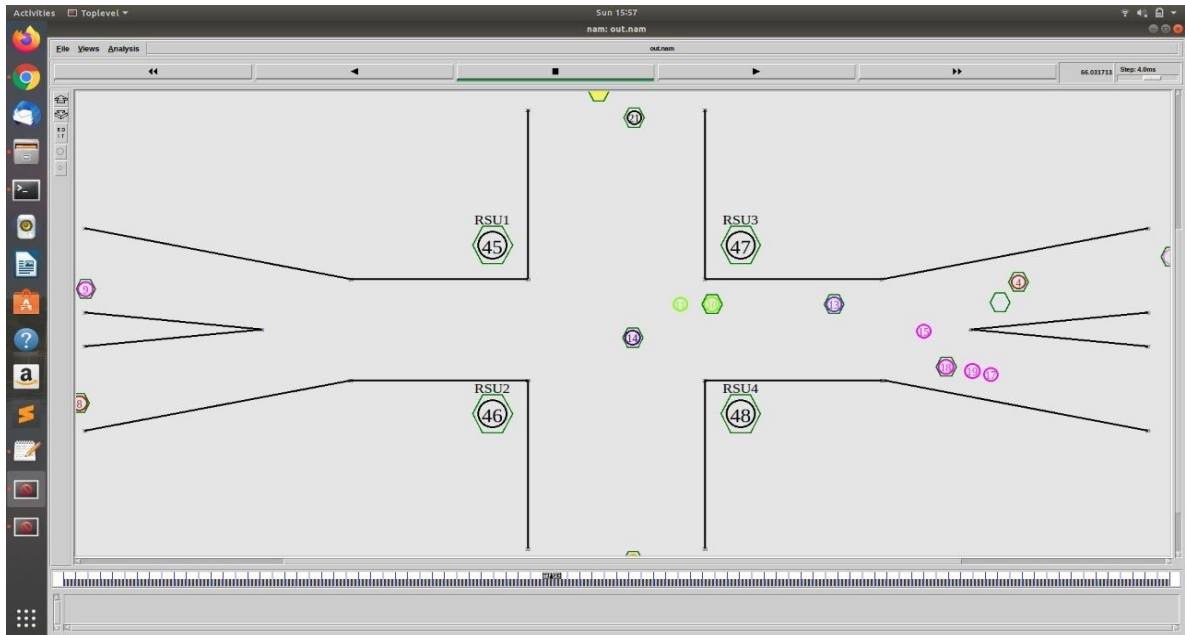


Figure 5.10: Output of Module-5 after merging

CM_changes.py

```
import matplotlib.pyplot as plt
f = open("NewMultihopPythonOutput.txt", "r")
total_file_list = f.readlines()
no_of_lines = len(total_file_list)
cm_head = [0] * 22
cm_head_1 = [0] * 22
global switches
switches = 0
prev_switches = 0
l1 = []
dur1=[]
dur2=[]
dur3=[]
def compare(aa,ch_flag, ch_flag_1):
```

```

        global switches
        for i in range(len(ch_flag)):
            if (ch_flag[i] != ch_flag_1[i]):
                switches += 1

for i in range(no_of_lines):
    cm_head=[0]*22
    line = total_file_list[i]
    line2=line[:len(line)-1]
    line2=line2.split("a")
    line = line.split("a")[1]
    line = (line.split(" "))
    no_of_clusters = line[0]
    line = line[1:len(line) - 1]
    ch_list=line
    cm_list=line2[2:]
    for k1 in range(len(cm_list)):
        cm_sublist=cm_list[k1]
        cm_sublist=(cm_sublist.split(" "))
        cm_sublist=cm_sublist[:len(cm_sublist) - 1]
        for cm in cm_sublist:
            cm_head[int(cm)]=ch_list[k1]
    for j in line:
        cm_head[int(j)] = -1
    compare(i,cm_head, cm_head_1)
    cm_head_1=cm_head
    if (i % 100 == 0):
        l1.append((switches - prev_switches))
        dur1.append(100/(switches-prev_switches))

```

```

        prev_switches = switches

f.close()
f = open("NHopPythonOutput.txt", "r")
total_file_list = f.readlines()
no_of_lines = len(total_file_list)
cm_head = [0] * 22
cm_head_1 = [0] * 22
switches = 0
prev_switches = 0
l2 = []
def compare(aa,ch_flag, ch_flag_1):
    global switches
    for i in range(len(ch_flag)):
        if (ch_flag[i] != ch_flag_1[i]):
            switches += 1

for i in range(no_of_lines):
    cm_head=[0]*22
    line = total_file_list[i]
    line2=line[:len(line)-1]
    line2=line2.split("a")
    line = line.split("a")[1]
    line = (line.split(" "))
    no_of_clusters = line[0]
    line = line[1:len(line) - 1]
    ch_list=line
    cm_list=line2[2:]
    for k1 in range(len(cm_list)):
        cm_sublist=cm_list[k1]
        cm_sublist=(cm_sublist.split(" "))

```

```

        cm_sublist=cm_sublist[:len(cm_sublist) - 1]
    for cm in cm_sublist:
        cm_head[int(cm)]=ch_list[k1]
    for j in line:
        cm_head[int(j)] = -1
    compare(i,cm_head, cm_head_1)
    cm_head_1=cm_head
    if (i % 100 == 0):
        l2.append((switches - prev_switches))
        dur2.append(100/(switches-prev_switches))
        prev_switches = switches

f.close()
l2=[k1+30 for k1 in l2]
f = open("SingleHopPythonOutput.txt", "r")
total_file_list = f.readlines()
no_of_lines = len(total_file_list)
cm_head = [0] * 22
cm_head_1 = [0] * 22
switches = 0
prev_switches = 0
l3 = []
def compare(aa,ch_flag, ch_flag_1):
    global switches
    for i in range(len(ch_flag)):
        if (ch_flag[i] != ch_flag_1[i]):
            switches += 1
for i in range(no_of_lines):
    cm_head=[0]*22
    line = total_file_list[i]

```



```

        line2=line[:len(line)-1]
        line2=line2.split("a")
        line = line.split("a")[1]
        line = (line.split(" "))
        no_of_clusters = line[0]
        line = line[1:len(line) - 1]
        ch_list=line
        cm_list=line2[2:]
    for k1 in range(len(cm_list)):
        cm_sublist=cm_list[k1]
        cm_sublist=(cm_sublist.split(" "))
        cm_sublist=cm_sublist[:len(cm_sublist) - 1]
        for cm in cm_sublist:
            cm_head[int(cm)]=ch_list[k1]
    for j in line:
        cm_head[int(j)] = -1
    compare(i,cm_head, cm_head_1)
    cm_head_1=cm_head
    if (i % 100 == 0):
        l3.append((switches - prev_switches))
        dur3.append(100/(switches-prev_switches))
        prev_switches = switches
f.close()
x = [10, 20, 30, 40, 50, 60, 70, 80, 90]
y1 = l1[1:10]
y2 = l2[1:10]
y3 = l3[1:10]
plt.plot(x, y3, color='black', linestyle='solid', linewidth = 1,marker='D',
markerfacecolor='black', markersize=6,label='1-hop')

```

```

plt.plot(x, y1, color='green', linestyle='solid', linewidth = 1,marker='*',
markerfacecolor='green', markersize=6,label='New')
plt.plot(x, y2, color='blue', linestyle='solid', linewidth = 1,marker='X',
markerfacecolor='blue', markersize=6,label='N-Hop')
plt.xlabel('Time')
plt.ylabel('Cluster Member Changes')
plt.title('Performance Analysis')
plt.legend()
plt.show()
y1 = dur1[1:10]
y2 = dur2[1:10]
y3 = dur3[1:10]
plt.plot(x, y3, color='black', linestyle='solid', linewidth = 1,marker='D',
markerfacecolor='black', markersize=6,label='1-hop')
plt.plot(x, y2, color='blue', linestyle='solid', linewidth = 1,marker='X',
markerfacecolor='blue', markersize=6,label='Nhop')
plt.plot(x, y1, color='green', linestyle='solid', linewidth = 1,marker='*',
markerfacecolor='green', markersize=6,label='New')
plt.xlabel('Time')
plt.ylabel('Average Cluster Member Duration')
plt.title('Performance Analysis')
plt.legend()
plt.show()

```

CH_Changes.py

```

import matplotlib.pyplot as plt
dur1=[]
dur2=[]
dur3=[]
f= open("NewMultihopPythonOutput.txt","r")

```

```

total_file_list=f.readlines()
no_of_lines=len(total_file_list)
ch_flag=[0]*22
ch_flag_1=[0]*22
global switches
switches=0
prev_switches=0
l1=[]
l2=[]
def compare(ch_flag,ch_flag_1):
    global switches
    for i in range(len(ch_flag)):
        if(ch_flag[i]!=ch_flag_1[i]):
            switches+=1
for i in range(no_of_lines):
    ch_flag = [0] * 22
    line=total_file_list[i]
    line=line.split("a")[1]
    line=(line.split(" "))
    line=line[1:len(line)-1]
    for j in line:
        ch_flag[int(j)]=1
    compare(ch_flag,ch_flag_1)
    ch_flag_1=ch_flag
    if(i%100==0):
        l1.append((switches-prev_switches))
        dur1.append(100/(switches-prev_switches))
        prev_switches = switches
f.close()

```

```

f= open("NHopPythonOutput.txt","r")
total_file_list=f.readlines()
no_of_lines=len(total_file_list)
ch_flag=[0]*22
ch_flag_1=[0]*22
switches=0
prev_switches=0
def compare(ch_flag,ch_flag_1):
    global switches
    for i in range(len(ch_flag)):
        if(ch_flag[i]!=ch_flag_1[i]):
            switches +=1
for i in range(no_of_lines):
    ch_flag = [0] * 22
    line=total_file_list[i]
    line=line.split("a")[1]
    line=(line.split(" "))
    line=line[1:len(line)-1]
    for j in line:
        ch_flag[int(j)]=1
    compare(ch_flag,ch_flag_1)
    ch_flag_1=ch_flag
    if(i%100==0):
        l2.append((switches-prev_switches))
        dur2.append(100/(switches-prev_switches))
        prev_switches=switches
f.close()
l2=[k1+50 for k1 in l2 ]
f= open("SingleHopPythonOutput.txt","r")

```

```

total_file_list=f.readlines()
no_of_lines=len(total_file_list)
ch_flag=[0]*22
ch_flag_1=[0]*22
switches=0
prev_switches=0
l3=[]
def compare(ch_flag,ch_flag_1):
    global switches
    for i in range(len(ch_flag)):
        if(ch_flag[i]!=ch_flag_1[i]):
            switches +=1
for i in range(no_of_lines):
    ch_flag = [0] * 22
    line=total_file_list[i]
    line=line.split("a")[1]
    line=(line.split(" "))
    line=line[1:len(line)-1]
    for j in line:
        ch_flag[int(j)]=1
    compare(ch_flag,ch_flag_1)
    ch_flag_1 = ch_flag
    if(i%100==0):
        l3.append((switches-prev_switches))
        dur3.append(100/(switches-prev_switches))
        prev_switches=switches
f.close()
x = [10, 20, 30, 40, 50, 60, 70, 80, 90]
y1 = l1[1:10]

```

```

y2 = l2[1:10]
y3 = l3[1:10]
plt.plot(x, y3, color='black', linestyle='solid', linewidth = 1,marker='D',
markerfacecolor='black', markersize=6,label='1-hop')
plt.plot(x, y2, color='blue', linestyle='solid', linewidth = 1,marker='X',
markerfacecolor='blue', markersize=6,label='Nhop')
plt.plot(x, y1, color='green', linestyle='solid', linewidth = 1,marker='*',
markerfacecolor='green', markersize=6,label='New')
plt.xlabel('Time')
plt.ylabel('Cluster Head Changes')
plt.title('Performance Analysis')
plt.legend()
plt.show()
y1 = dur1[1:10]
y2 = dur2[1:10]
y3 = dur3[1:10]
plt.plot(x, y3, color='black', linestyle='solid', linewidth = 1,marker='D',
markerfacecolor='black', markersize=6,label='1-hop')
plt.plot(x, y2, color='blue', linestyle='solid', linewidth = 1,marker='X',
markerfacecolor='blue', markersize=6,label='Nhop')
plt.plot(x, y1, color='green', linestyle='solid', linewidth = 1,marker='*',
markerfacecolor='green', markersize=6,label='New')
plt.xlabel('Time')
plt.ylabel('Average Cluster Head Duration')
plt.title('Performance Analysis')
plt.legend()
plt.show()

```

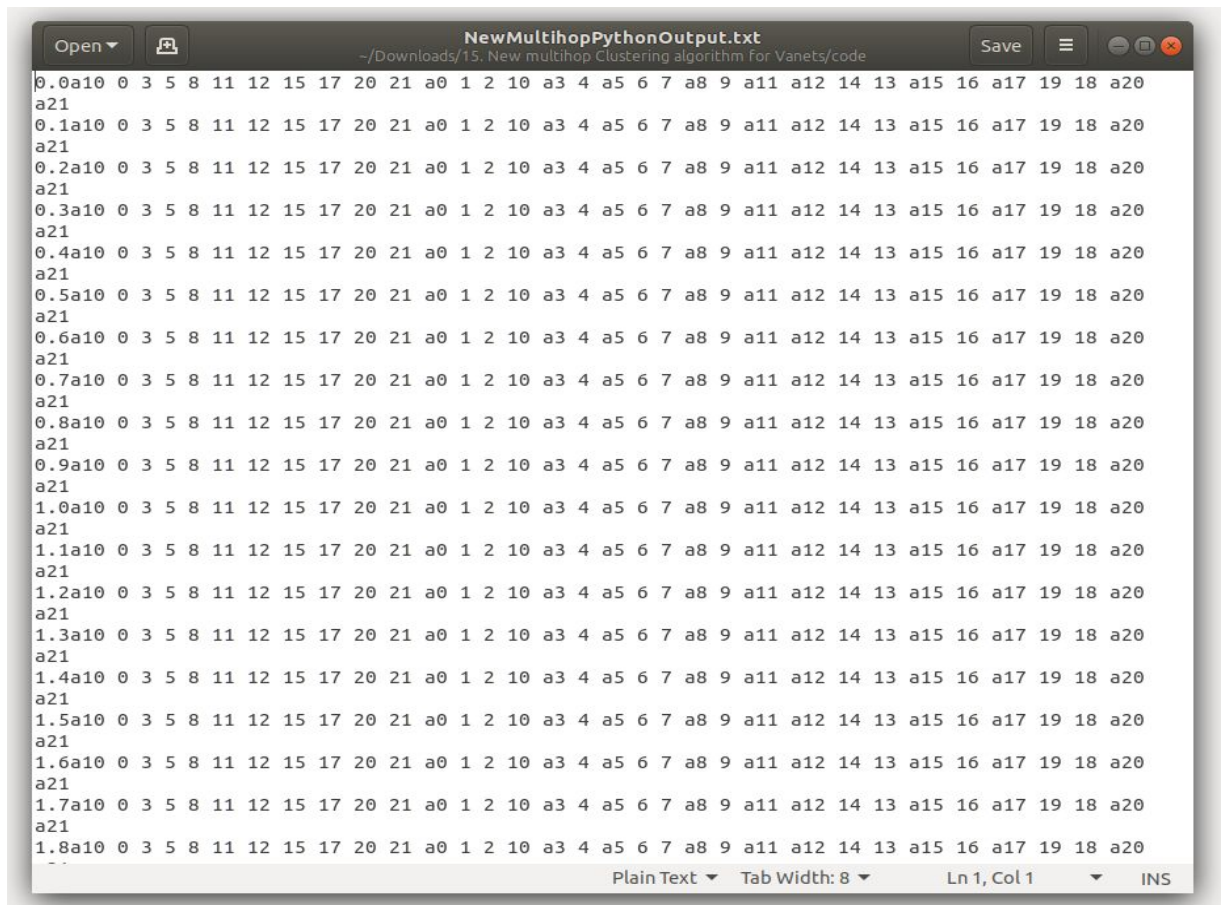
Chapter 6

Experimental Results And Analysis

After running the script it generates a NAM trace file that is going to be used as an input to NAM and a trace file is called out.tr that will be used for our simulation analysis.

6.1 Output Format

In the output of our algorithm, first it contains the time instance. Next it contains the number of cluster heads present. Later followed by the cluster head and next it shows the details of the cluster (cluster head and the cluster members of the cluster).



```
0.0a10 0 3 5 8 11 12 15 17 20 21 a0 1 2 10 a3 4 a5 6 7 a8 9 a11 a12 14 13 a15 16 a17 19 18 a20
a21
0.1a10 0 3 5 8 11 12 15 17 20 21 a0 1 2 10 a3 4 a5 6 7 a8 9 a11 a12 14 13 a15 16 a17 19 18 a20
a21
0.2a10 0 3 5 8 11 12 15 17 20 21 a0 1 2 10 a3 4 a5 6 7 a8 9 a11 a12 14 13 a15 16 a17 19 18 a20
a21
0.3a10 0 3 5 8 11 12 15 17 20 21 a0 1 2 10 a3 4 a5 6 7 a8 9 a11 a12 14 13 a15 16 a17 19 18 a20
a21
0.4a10 0 3 5 8 11 12 15 17 20 21 a0 1 2 10 a3 4 a5 6 7 a8 9 a11 a12 14 13 a15 16 a17 19 18 a20
a21
0.5a10 0 3 5 8 11 12 15 17 20 21 a0 1 2 10 a3 4 a5 6 7 a8 9 a11 a12 14 13 a15 16 a17 19 18 a20
a21
0.6a10 0 3 5 8 11 12 15 17 20 21 a0 1 2 10 a3 4 a5 6 7 a8 9 a11 a12 14 13 a15 16 a17 19 18 a20
a21
0.7a10 0 3 5 8 11 12 15 17 20 21 a0 1 2 10 a3 4 a5 6 7 a8 9 a11 a12 14 13 a15 16 a17 19 18 a20
a21
0.8a10 0 3 5 8 11 12 15 17 20 21 a0 1 2 10 a3 4 a5 6 7 a8 9 a11 a12 14 13 a15 16 a17 19 18 a20
a21
0.9a10 0 3 5 8 11 12 15 17 20 21 a0 1 2 10 a3 4 a5 6 7 a8 9 a11 a12 14 13 a15 16 a17 19 18 a20
a21
1.0a10 0 3 5 8 11 12 15 17 20 21 a0 1 2 10 a3 4 a5 6 7 a8 9 a11 a12 14 13 a15 16 a17 19 18 a20
a21
1.1a10 0 3 5 8 11 12 15 17 20 21 a0 1 2 10 a3 4 a5 6 7 a8 9 a11 a12 14 13 a15 16 a17 19 18 a20
a21
1.2a10 0 3 5 8 11 12 15 17 20 21 a0 1 2 10 a3 4 a5 6 7 a8 9 a11 a12 14 13 a15 16 a17 19 18 a20
a21
1.3a10 0 3 5 8 11 12 15 17 20 21 a0 1 2 10 a3 4 a5 6 7 a8 9 a11 a12 14 13 a15 16 a17 19 18 a20
a21
1.4a10 0 3 5 8 11 12 15 17 20 21 a0 1 2 10 a3 4 a5 6 7 a8 9 a11 a12 14 13 a15 16 a17 19 18 a20
a21
1.5a10 0 3 5 8 11 12 15 17 20 21 a0 1 2 10 a3 4 a5 6 7 a8 9 a11 a12 14 13 a15 16 a17 19 18 a20
a21
1.6a10 0 3 5 8 11 12 15 17 20 21 a0 1 2 10 a3 4 a5 6 7 a8 9 a11 a12 14 13 a15 16 a17 19 18 a20
a21
1.7a10 0 3 5 8 11 12 15 17 20 21 a0 1 2 10 a3 4 a5 6 7 a8 9 a11 a12 14 13 a15 16 a17 19 18 a20
a21
1.8a10 0 3 5 8 11 12 15 17 20 21 a0 1 2 10 a3 4 a5 6 7 a8 9 a11 a12 14 13 a15 16 a17 19 18 a20
a21
```

Figure 6.1: Output Format

6.2 Performance Analysis:

The performance of the New Clustering algorithm is measured against the existing Nhop and Single Hop algorithms.

The performance of the algorithm is done by measuring the

1. Average Cluster Head Changes
2. Average Cluster Member Changes
3. Average Cluster Head Duration
4. Average Cluster Member Duration

Stability $\propto 1/\text{ClusterHead(Member) changes}$

Stability $\propto \text{ClusterHead(Member) Duration}$

Plots of these performance analysis are done using python.

6.2.1 Time vs Cluster Head Changes

This graph shows us the variation of cluster head changes with the time. In the singlehop and Nhop algorithms there are more number of cluster head changes and in our new algorithm there are less number of cluster head changes leading to the fact that our algorithm is better when compared to other two algorithms. This graph clearly depicts that our New Multihop Clustering Algorithm is much better than those traditional Nhop algorithm and Single algorithm.

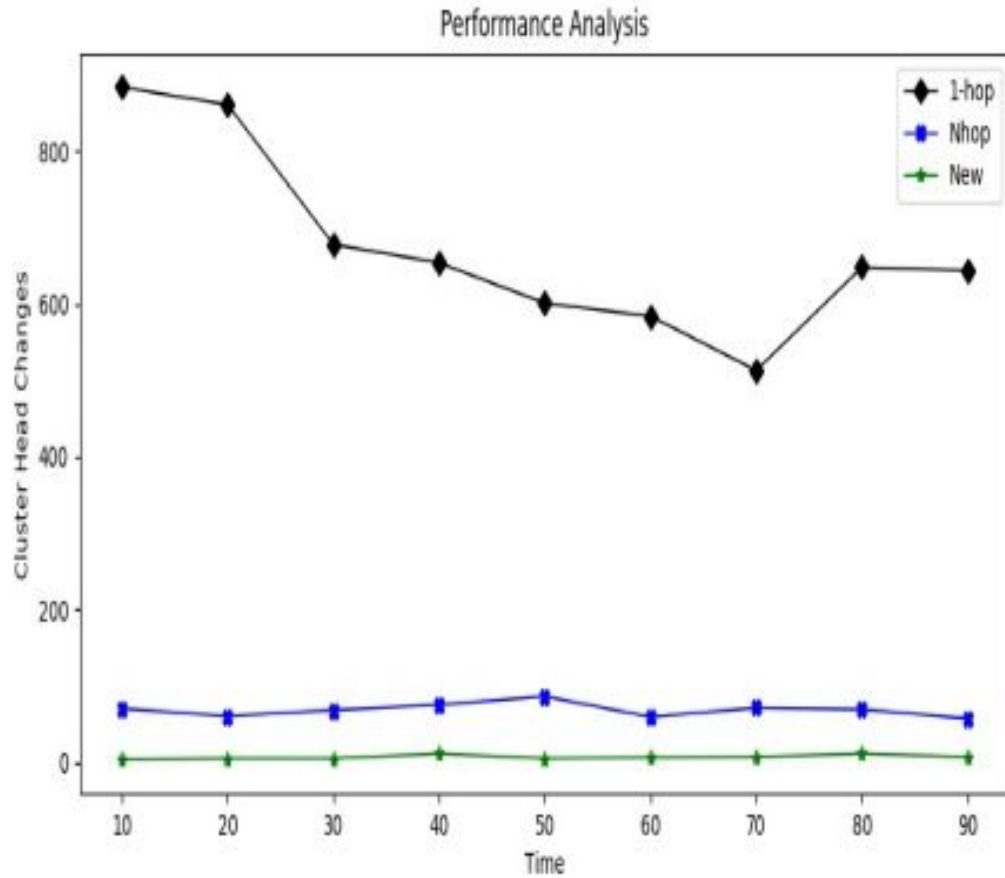


Figure 6.2: Time vs Cluster Head Changes

6.2.2 Time vs Average Cluster Head Duration

This graph shows us the variation of cluster head durations with the time. As said above, there are more number of cluster head changes in singlehop and Nhop leads to less duration to the cluster head. Our new algorithm has more duration of cluster head leading to formation of stable clusters.

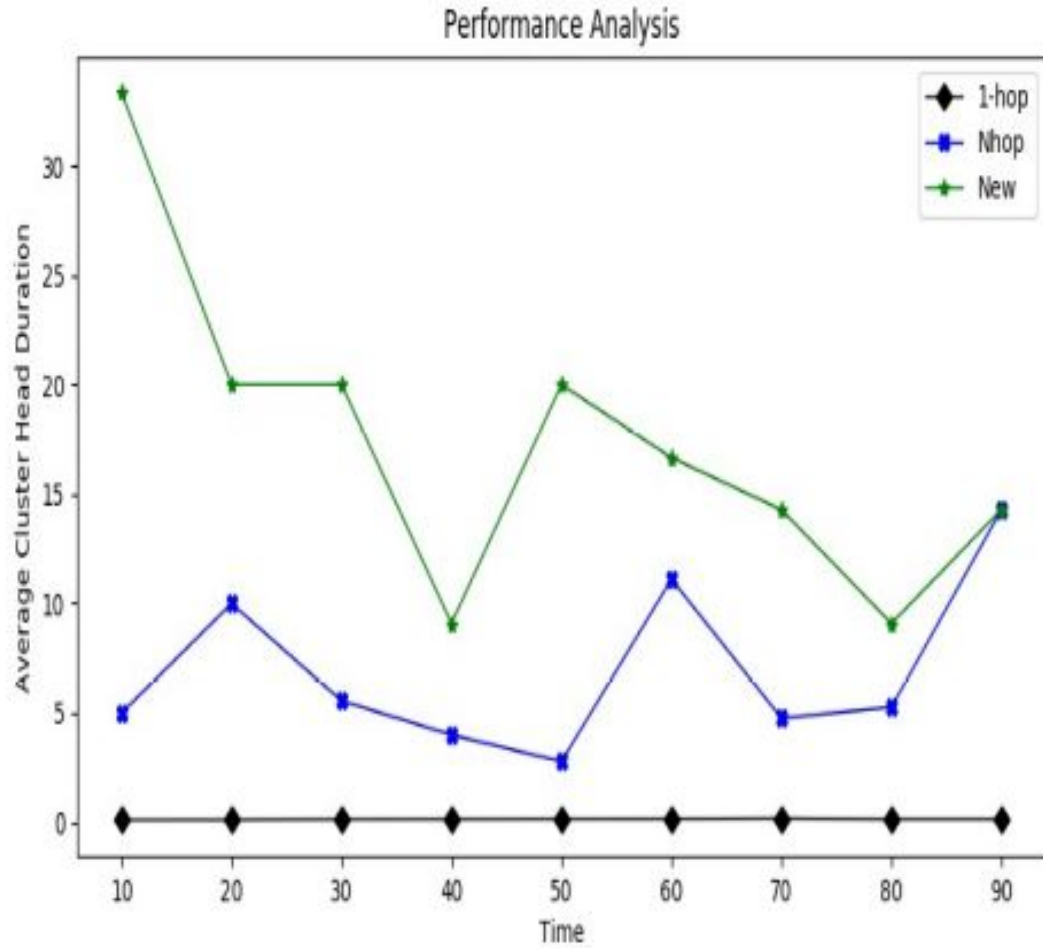


Figure 6.3: Time vs Average Cluster Head Duration

6.2.3 Time vs Cluster Member Changes

This graph shows us the variation of cluster member changes with the time. In the singlehop and Nhop algorithms there are more number of cluster member changes and in our new algorithm there are less number of cluster member changes leading to the fact that our algorithm is better when compared to other two algorithms.

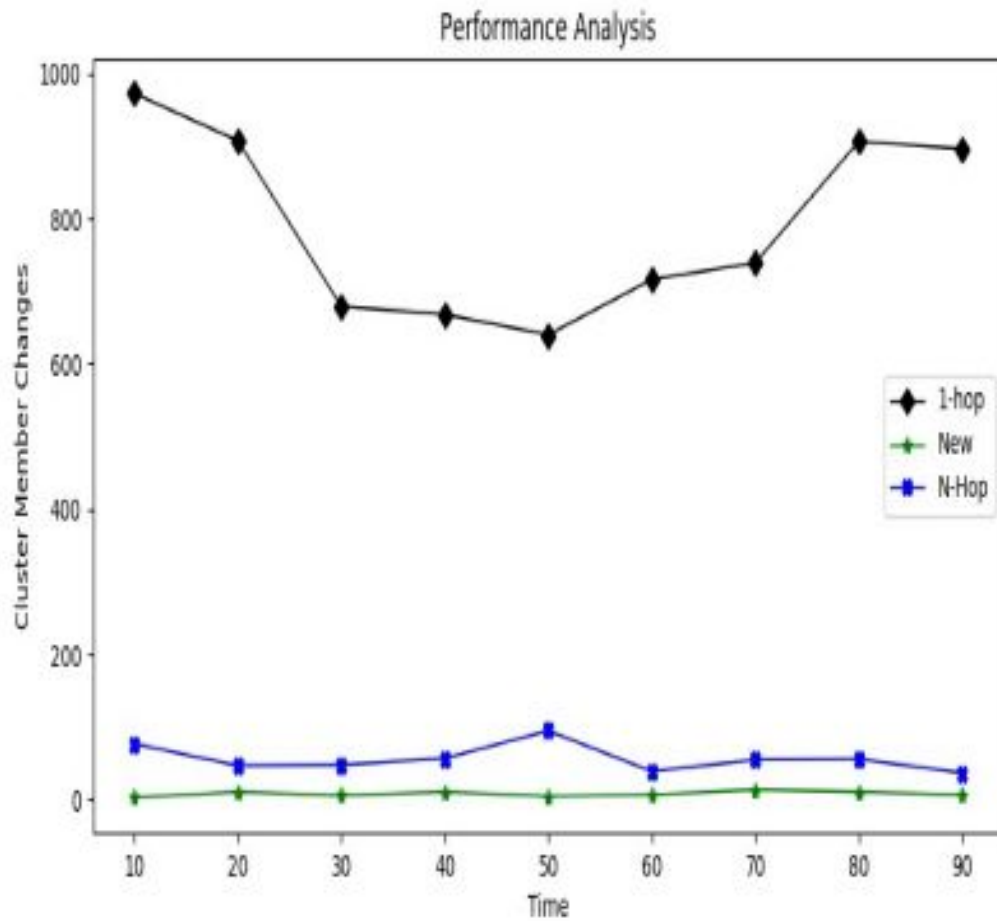


Figure 6.4: Time vs Cluster Member Changes

6.2.4 Time vs Average Cluster Member Duration

This graph shows us the variation of cluster member durations with the time. As said above, there are more number of cluster member changes in singlehop and Nhop leads to less duration to the cluster member.

Our new algorithm has more duration of cluster members as there are less number of cluster member changes.

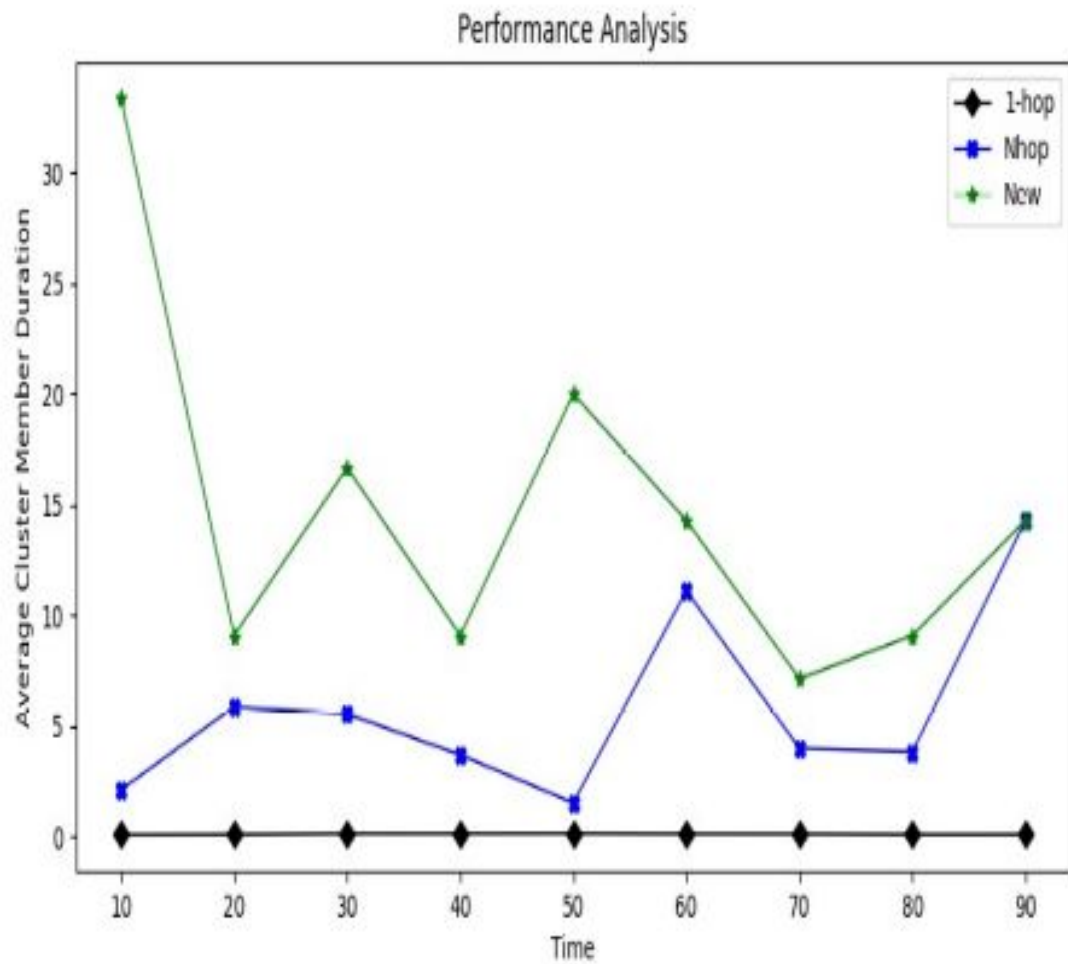


Figure 6.5 : Time vs Average Cluster Member Duration

Chapter 7

Conclusion And Scope of Future Work

7.1 Conclusion :

This chapter gives the summary of the research work done and highlights the research contribution of the thesis and suggests directions for possible future work. We obtain the neighbouring nodes info like speed and direction. Then we form the most stable cluster and find the best cluster head in the cluster. Based on our experiments, we can see that our proposed algorithm can improve the stability and reliability of VANET.

7.2 Future work :

It's a bit complex algorithm and can simplify further. By using distributed clustering algorithm strategies we can still improve our algorithm. In our algorithm the cluster merging is done when one cluster interferes with the other. An optimal inter cluster merging strategies can be used to further increase the cluster performance.

We can use a double headed clustering algorithm for more stability.

Chapter 8

References

- [1] Degan Zhang, Hui Ge, Ting Zhang, Yu-Ya Cui, “New Multi-Hop Clustering Algorithm for Vehicular Ad Hoc Networks”, IEEE Transactions on Intelligent Transportation Systems, Volume:20 Issue:4 April 2019.

- [2] Yuzhong Chen, Mingyue Fang, Song Shi, Wenzhong Guo & Xianghan Zheng, “Distributed multi-hop clustering algorithm for VANETs based on neighborhood follow”, EURASIP Journal on Wireless Communications and Networking 2015.

- [3] P.Deepthi, S.Sivakumar, R. Murugesan, “Development of Multihop clustering algorithm for the simulation of VANET”, International Journal of Computational and Applied Mathematics. ISSN 1819-4966, Volume 12, Number 1 (2017).

- [4] Meysam Azizian, Soumaya Cherkaoui, Abdelhakim Senhaji Hafid, “A distributed D-hop cluster formation for VANET”, IEEE Wireless Communications and Networking Conference 2016.

- [5] Rajan Jamgekar, Snehal Tapkire, “A robust multi-hop clustering algorithm for reliable VANET message dissemination”, 2017 International Conference on energy, Communication, Data Analytics and Soft Computing(ICECDS), ISBN: 978-1-5386-1887-5, pp.2599-2604, 2018.

- [6]** Meysam Azizian, Soumaya Cherkaoui, Abdelhakim Senhaji Hafid, “A distributed cluster formation for VANET based on end-to-end relative mobility”, IEEE Wireless Communications and Mobile Computing Conference(IWCMC), ISBN: 978-1-5090-0304-4, pp.287-291, 2016.
- [7]** Chulhee Jang, Jae Hong Lee, “Path Selection Algorithms for Multi-Hop VANET”, IEEE 72nd Vehicular Technology Conference – Fall Sept 2010.
- [8]** Mahmoud A.Alawi, Rashid, A.Saeed, “Cluster-based multi-hop vehicular communication with multi-metric optimization”, International Conference on Computer and Communication Engineering(ICCCE) July 2012 .
- [9]** Samo Vodopivec, Janez Bester, “A survey on clustering algorithms for vehicular ad-hoc networks”, 2012 35th International Conference on Telecommunications and Signal Processing(TSP), ISBN: 978-1-4673-1118-2, pp.52-56, 2012.
- [10]** Oussama Senouci, Zibounda Aliouat, Saad Harous, “A Multi-hop Clustering Approach over Vehicle-to-Internet communication for improving VANETs performances”, Future Generation Computer Systems Volume 96, July 2019, pp.309-323, 21 February 2019.
- [11]** Soufiane Ouahou, Slimane Bah, Zohra Bakkoury, “Multi-hop Clustering Solution Based on Beacon Delay for Vehicular Ad-Hoc Networks”, International Symposium on Ubiquitous Networking, pp. 357-367, 8 November 2017.

[12] Sanaz Khakpour, “A distributed clustering algorithm for target tracking in vehicular ad-hoc networks”, 3rd ACM international symposium on Design and analysis of intelligent vehicular networks and applications, pp.145-152, November 2013.

[13] Zhenxia Zhang, Azzedine Boukerche, “A novel multi-hop clustering scheme for vehicular ad-hoc networks”, ACM international symposium on Mobile management and wireless access, pp.19-26, October 2011.

[14] Ahmed Alioua, Sidi-Mohammed Senouci, “A Distributed Multi-hop Clustering Algorithm for Infrastructure-Less Vehicular Ad-Hoc Networks” , Lecture Notes of the Institute for Computer sciences, Social Informatics and Telecommunications Engineering book series (LNICST, volume 244) pp.68-81.

[15] Jyotsna rao Dawande, Sanjay Silakari, “Enhanced Distributed Multi-Hop Clustering Algorithm for VANETs Based on Neighborhood Follow (EDMCNF) Collaborated with Road Side Units”, 2015 International Conference on Computational Intelligence and Communication Networks (CICN) , ISSN: 2472-7555, pp.106-113, 2015.

[16] Seyhan Ucar, Sinem Coleri Ergen, “VMaSC: Vehicular multi-hop algorithm for stable clustering in Vehicular Ad Hoc Networks”, 2013 IEEE Wireless Communications and Networking Conference (WCNC), ISBN: 978-1-4673-5939-9, pp.2381-2386, Shanghai, China , 2013.

[17] Ke Huang, Bin-Jie Hu, “A New Distributed Mobility-Based Multi-Hop Clustering Algorithm for Vehicular Ad Hoc Networks in

Highway Scenarios”, 2019 IEEE 90th Vehicular Technology Conference(VTC2019-Fall), ISBN: 978-1-7281-1220-6, pp.1-6, 2019.

[18] Craig Cooper, Daniel Franklin, “A Comparative Survey of VANET Clustering Techniques”, IEEE Communications Surveys & Tutorials, Volume: 19, Issue: 1, pp.657-681, 2016.

[19] Sarah Oubabas, Rachida Aoudjit, “Secure and stable Vehicular Ad Hoc Network clustering algorithm based on hybrid mobility similarities and trust management scheme”, Vehicular Communications, Volume 13, pp.128-138, 2018.

[20] Ahmed Louazani, Sidi Mohammed Senouci, “Clustering-based algorithm for connectivity maintenance in Vehicular Ad-Hoc Networks”, 2014 14th International Conference on Innovations for Community Services(14CS), ISBN: 978-1-4799-5350-9, papers (13), 2014.

[21] Poonam Thakur, “A Comparative Study of Cluster-Head Selection Algorithms in VANET ”, Lecture Notes in Networks and Systems(LNNS), volume 116, 2010.

[22] Xianlei Ge, Qiang Gao, “A Novel Clustering Algorithm Based on Mobility for VANET”, 2018 IEEE 18th International Conference on Communication Technology(ICCT), ISBN: 978-1-5386-7635-6, pp.473-477, 2018.

[23] Kang Tan, Julien Le Kernec, “Clustering Algorithm in Vehicular Ad-hoc Networks: A Brief Summary”, 2109 UK/China Emerging Technologies(UCET), ISBN: 978-1-7281-2797-2, pp.1 - 5, 2019.

[24] Mengying Ren, Lyes Khoukhi, Jun Zhang, “A new mobility-based clustering algorithm for vehicular ad hoc networks (VANETs)” , NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium, ISBN: 978-1-5090-0223-8 , pp.1203-1208, 2016.