1-1-2013

Performance Evaluation Of Voice Chat In Vechicular Ad-Hoc Networks

Vineeth Rakesh Mohan
Wayne State University,

Recommended Citation

Mohan, Vineeth Rakesh, "Performance Evaluation Of Voice Chat In Vechicular Ad-Hoc Networks" (2013). *Wayne State University Theses.* Paper 283.

PERFORMANCE EVALUATION OF VOICE CHAT IN VEHICULAR AD-HOC NETWORKS

by

VINEETH RAKESH MOHAN

THESIS

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

2013

MAJOR: COMPUTER SCIENCE

Approved by:

Advisor Date

TABLE OF CONTENTS

LIST OF TABLES

Table

LIST OF FIGURES

Illustration

1 VANET COMMUNICATION

Vehicular ad-hoc networks (VANET) technology is bound to change the way we interact with vehicles. The two main applications that drive the VANET technology are safety and comfort. Safety applications involve real time traffic updates and updates on vehicular collisions while comfort applications involve applications like voice chat, real-time gaming, video chat etc. Research on safety applications has been widely explored, but there are very few studies that talk about VANET comfort applications. In this Thesis, we develop a voice chat model for V2V communication that can enable users in a vehicle to talk to other users in multiple vehicles that travel within certain distance from each other.

The communication in vehicular networks can be categorized into two types: vehicle to vehicle communication (V2V), and vehicle to infrastructure communication (V2I). The V2V involves communications between the vehicles forming an ad-hoc network. V2I communication is highly useful for applications like: point-of-interest notifications, podcasting, multi-hop wireless Internet access, and road-side video advertisement broadcasting. The V2V communication draws more attention from the VANET community since V2V communication meets the low latency requirements for safety applications. The V2V communication also eliminates the necessity to install Base Stations in order to maintain the connectivity; thereby, reducing the cost factor. Due to such advantages, we consider V2V as the mode of communication between the vehicles.

To implement our voice chat model, we use the popular NS2 network simulator. NS2 is one of the widely used simulation environment for VANET and other network simulations which is capable of supporting complex and time-consuming simulations.

However, the trace generated by NS2 does not produce realistic movement of vehicles. Therefore, we used the microscopic traffic simulator VISSIM [1] to generate trace that replicates the real-time movement of vehicles on a freeway. VISSIM is widely used in many industries to model real road systems. Throughout our simulation we use Flood and Scalable broadcast protocol (SBA) broadcast protocols to route the data packets and study their performance over the voice chat model.

1.2 MOTIVATION

Due to growing demands in inter-vehicle entertainment many automobile companies are investing time and money in VANET comfort applications. There are a large number of studies that propose innovative concepts in V2V and V2I communication. However, most research tend to confine themselves towards security issues or connectivity quality between vehicles [2-4]. This has been a motivating factor for us to bring out this Thesis, which proposes a voice chat model for V2V communication. We chose voice chat as the entertainment application since it is the most widely preferred application.

1.3 GOAL OF THE THESIS

The goal of this thesis is defined as follows:

1. Explore the feasibility of group chat, and the nature of user behavior in VANET communication.
2. Run extensive simulations on real-time traffic models using NS2 simulator to mimic the real world voice chat scenario.

3. Compare the performance voice chat application using Flooding broadcast and Scalable broadcast algorithm (SBA).

1.3 TASK DEFINITION

The scope of this thesis is to develop a voice chat model and validate it using NS2 simulations. We divide the task into following subtasks:

- **Develop a Voice Chat Algorithm**

Our primary goal is to develop a voice chat algorithm that can mirror an actual voice chat behavior. To achieve this goal, group chat parameters like group size, talk spurt period and type of distribution are modeled according to a real-time group chat model [5].

- **Integrating VANET code to NS2 Environment**

The experiments were run using the NS-2.34 version of NS2. It was necessary to modify existing VANET code for earlier versions of NS2 to suite the latest NS-2.34 environment. This task was a critical part of the simulation since underlying VANET code forms the base for rest of the simulation.

- **Implementing the Broadcast Protocol in NS2**

In this Thesis, Flooding and Scalable Broadcast were chosen as the broadcast algorithms to effectively broadcast the voice packets. Implementation of these protocols was an essential part of this thesis. We borrowed the implementation from BCAST [6], which is a modified version of the Scalable Broadcast algorithm. The BCAST protocol code was extensively modified to suite our application.

- **Generating Real-time VANET Trace**

Since simulating the voice chat model involving a real-time vehicular trace contributes a major part of the thesis, the real-time vehicular traffic trace on I-75S freeway was generated using the microscopic VISSIM simulator by Rao et al. [7] specifically for this Thesis.

- **Extensive Testing of the Simulation**

The simulation developed as a part of this thesis has to be extensively tested by using large simulation times. For this reason the simulation has been tested over 10 randomly selected vehicle groups. In the first part of simulation there are 5 group members (vehicles) in one group, and they from the group for a casual voice chat. The experiments are repeated for a group size of 3 again. The simulations are separately executed for two different transmission ranges of 200 and 600 meters and varying densities of 250 vehicles per hour (vph) and 1200vph. So the results of this thesis are the outcome of about 720 individual NS2 simulations with the simulation time set to 1800sec. On an average the actual running time for each simulation turned out to be about a day and half. The entire simulation was run on WSU grid.

1.5 DOCUMENT STRUCTURE

This Thesis is divided into 12 sections. Section 2 defines some fundamental concepts that are needed to understand the rest of the document. Section 3 talks about other works related to VANET communication. Section 4 explains the implementation of simulation, which is divided into 3 subsections; the first subsection presents the group chat algorithm and its implementation in NS2; the second one describes the implementation of broadcast technique in NS2, and the final subsection talks about the trace generation process. Section 5 through 10 presents the results of our experiment. In Section 5, we show the results of content delivery using Flood and SBA broadcast techniques; Section 6 gives the delay statistics; in Sections 7 and 8 we present the results of group talk patterns and fairness patterns respectively; Section 9 describes the statistics on interrupted nodes; Section 10 compares the jitter delays of Flood and SBA broadcast and Section 11 presents further results to show that SBA underperforms in our application when compare to Flood. Finally, Section 12 describes the future work and Section 13 concludes this Thesis.

2 FUNDAMENTAL CONCEPTS

This section of the document explains some fundamental concepts and definitions that are used throughout this work. These concepts are essential for further understanding of this Thesis. Section 2.1 gives a briefing on the NS2 discrete event simulator; Section 2.2 explains the VANET extension that is developed for NS2; Section 2.3 talks about group chat behavior and some of the important parameters associated with it. The broadcast protocols are explained in Section 2.4, and Section 2.5 explains the real-time trace that is used for the experiment.

2.1 NS2 - A DISCRETE EVENT SIMULATOR

Simulation is the imitation of action of real systems and its development over time. It is a widely used methodology, which is used to closely model a real world system. A simulation consists of variables and assumptions for its operation. The variables correspond to the value of interest; for example, it can be the number of packets received. A simulator is called discrete-event simulator when changes in the model happen in distinct moments of time. The NS2 is a discrete event simulator which is widely used in the simulation of computer networks. These moments are defined and calculated by the model. Changes do not occur between two consecutive moments. Discrete-event simulations are used to simulate computer networks, since they have a discrete nature and they progress stepwise. The Algorithm of a discrete event simulator is described in figure 2.1.

Event

The event is the common term used in a discrete event simulation. An event is nothing but an action that is instructed to a simulation model at a given moment of time. An event can change the model variables and schedule another event(s) that happen as a consequence. The NS2 has been written in C/C++ and the TCL script is more like a wrapper. The events are signaled through the TCL scripts. For example, the *setdest* event produces the movement for the nodes in a wireless simulation scenario. By setting the values for x, y and z co-ordinates the node starts moving towards the specified position. The *send data* event is used to send the packet. Once the data packet is sent the packet is marked as sent, and the NS2 schedules a *received data* event after a period of time. The model is also capable of calculating the packet transmission time using its assumptions.

Simulation Time

Simulation time is the virtual time that is used to run the simulation model. The simulation time in NS2 proceeds step wise since there are no changes occurring between two consecutive events. It jumps over the moments at which the events happen. The events are usually stored in a list ordered by their timestamp. The simulator takes the next earliest event from the list and advances its virtual simulation time to the event’s timestamp and executes the event.

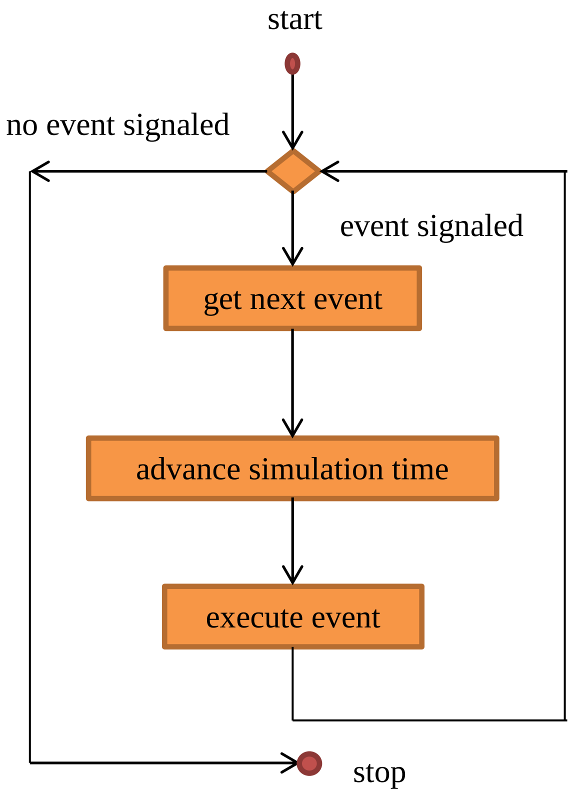


Figure 2.1 Discrete Event Simulator

Random Variables

The final abstraction that is common in simulation models is the use of random variables. The random variables are important numbers for any simulation because many of the actions in a simulation model involve a random phenomenon. In our voice chat model, we generate random variables from Weibull distribution to mirror the conventional group talk environment. Another classical example is the time when the simulation model performs the task. These are modeled by random variables following a uniform distribution with a mean of μ and variance σ^2. The pseudo random numbers are generated using random number generators (RNG). A RNG is an automaton that produces a sequence of numbers that tend to be random. To model this sequence we have to set a seed value. If the same seed value is used twice then the RNG produces the same sequence of random numbers. However, when the seed is changed; the sequence of random number changes, which then leads to different simulation results.

Relationship between Events

It is necessary to briefly understand the working of events. An event A effects B if the following conditions are met

1. A occurs before B and
2. A modifies the variable that B uses

This clearly means that B depends on$A$. It must also be mentioned that the relationship is transitive in nature; i.e., if A affects another event C and C affects B then A affects B. On the other hand if A does not affect B and B does not affect A then they both are independent.

In NS2 there are many dependent events hence the simulation model must be carefully designed and calibrated to successfully mirror the real world application that is being developed.

2.2 VANET EXTENSION FOR NS2

The VANET extension code was implemented over the existing code developed by [7]. The important modules implemented in the VANET extension are as follows

- **Neighborhood Table Module**: This module stores the location of each vehicle, the sequence number of the received packets, node id and the vehicle occupant.
- **Periodic Broadcast**: The periodic broadcast module is used to periodically send updates from each vehicle. These packets contain information about the vehicle's location, id and other relevant parameters.
- **The Publish and Receive Module**: The vehicles involved in the group talk publish and receive logical messages through these modules

2.3 FORMULATING A GROUP TALK SCENARIO

The group talk algorithm mimics a casual talk that happens in a group of people. To develop this algorithm, the group talk specific parameters like: the number of people involved in a group, the talk time of each person, the silence duration of each person, and the total conversation time are modeled. Papp.et.al [8] studied the effect of group sizes on conversation. Although the group sizes varied from 2 to 24 members, the optimal group size for conversation was reported as 5. This means that most frequent conversations involved a group of 5 people. Hence, to evaluate the performance of our voice chat model we chose group sizes of 3 and 5.

Talkspurts and *Silence Periods* are the two main parameters to be considered. The *Talkspurts* signify the duration of each talk burst by a person involved in the group conversation, and the *Silence Periods* signify the duration for which the person does not talk. In short, a *Talkspurt* means ON and *Silence Period* implies OFF. Han.et.al [5] found that a casual group conversation follows Weibull distribution. Their model shows that 90% of the *Talkspurts* are shorter than 11 seconds and 90% of the *Silence Periods* are shorter than 117 seconds. Table 2.3 shows the value for *Scale Parameter* (ʎ) and *Shape Parameter* (k).

Group Talk Parameter	Value
Talkspurt	ʎ = 7.6, k = 2.6
Silence	ʎ = 58.0, k = 1.7

Table 2.3: Group Talk Specific Parameters

The final parameter that is being considered is the *Fairness Value* of the person. The *Fairness Value* is defined as the degree of claim to speak. Higher *Fairness Value* implies higher the chance of a person to dominate the talk duration in a group. We set a base *Fairness Value* of 20 for all the nodes involved in the group talk.

2.4 BROADCAST PROTOCOLS IN VANET

Many broadcast protocols have been proposed for VANET, they are categorized into Probability based method, Area based method and Neighbor knowledge method. Neighbor knowledge methods are frequently used for broadcasting in VANET due to its advantages over other methods. In this thesis, we evaluate our voice chat model using the basic Flooding broadcast protocol and one of the widely discussed Neighbor knowledge broadcast protocol called the Scalable Broadcast Algorithm (SBA). The functioning of these broadcast protocols is described below.

2.4.1 BROADCAST BY FLOODING

The Flood broadcast protocol is the basic broadcast protocol for MANET communication. As proposed by [9], if there are n nodes in a group, each node attempts to broadcast m messages to all other nodes in the network. Therefore, the application expects $m * (n - 1)$ messages. When the nodes receive a packet it waits for a time interval that is uniformly distributed between 0 and the flooding interval before it rebroadcasts the packet. This time interval is set to $10ms$ in this simulation. Every node is equipped with a radio which is used for sending and receiving the packets. The range of this radio must be appropriately set to achieve efficacious broadcasting. An interesting observation by [9] show that the power range significantly affects the packet loss. A low power range can result in high packet loss since the disconnectivity between the nodes increase. However, with high ranges the collisions between the nodes are very high. The optimal range was observed between 150-250meters. In this thesis, the experiments have been performed using two different ranges; a low range of 250m and a high range of 600m.

2.4.2 BROADCAST STORM PROBLEM

Since Flooding broadcast is used in the simulation, it is important to describe the Broadcast Storm Problem [10]. The Broadcast Storm has been one of the most fundamental problems in Flooding broadcast, which leads to three main problems. First, there is serious contention because the nodes or vehicles that rebroadcast the packet may be close to each other. Second, there is heavy redundancy because a physical location would be covered by transmission range of several hosts so a single packet may be received multiple times due to Broadcast. Third, there are higher chances of collisions since the RTS/CTS mechanism is inapplicable, and the timings of the rebroadcasts are highly correlated. These problems are collectively referred to as Broad Cast Storm problem.

Effect of Rebroadcasts

The effect of redundant broadcasts is definitely a serious problem; firstly a rebroadcast can provide only 61% of additional coverage over what has been covered already. Also, after the initial broadcast a rebroadcast can cover only 41% of additional area. It is also shown that if more than 4 redundant messages are heard by the hosts, then the additional area of coverage drops down to a critical low value.

The effect of redundant packets can be described using the Figures 2.4.1 and 2.4.2 [10]. From the Figure 2.4.1, we can see that at most 2 transmissions are needed for the green node to broadcast the packet to all the other nodes. Nonetheless, if the redundant broadcasts are not controlled, it would result in 7 transmissions, which is the typical effect of flooding.

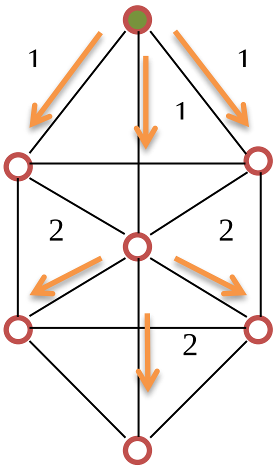


Figure 2.4.1 Number of Rebroadcasts

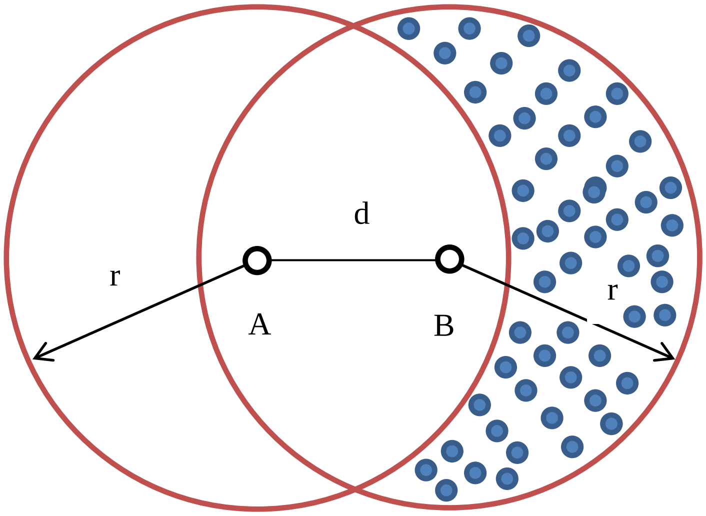


Figure 2.4.2 Effect of Rebroadcast

Tsung.et.al [10] provide some important equations that show the extent to which the area covered drops down with the increase in the number of redundant broadcasts. We present their study in brief to stress the importance of efficient broadcast mechanism. In Figure 2.4.2, node A sends a broadcast message to node B, and node B tries to rebroadcast the packet. The areas covered by nodes A and B are S_A and S_B respectively. We are concerned about the additional area that can be covered by $B's$ rebroadcast of the packet. The shaded region in the figure represents the area that would be covered by $B's$ rebroadcast. The maximum area that would be covered by $B's$ rebroadcast is given by the equation S_{B-A}. The S_{B-A} is derived to be equal to $S_B - S_{A \cap B} = \pi r^2 - INTR(d)$. Where the $INTR(d)$ denotes the intersection of the two circles separated by a distance d. On an average, the equation $\pi r^2 - INTR(d)$ turns out to be approximately equal to $0.41\pi r^2$. This means that the rebroadcast would only cover 41% percentage of the node's area of coverage. This effect becomes even more serious if the broadcast message is heard twice by a node. Consider a node Z that receives the same broadcast packet from node X and Y;

the additional area covered by $Z's$ rebroadcast would be S_{Z-XUY}. The value of S_{Z-XUY} was calculated as $0.19\pi r^2$, which is just 19% of additional area. Furthermore, if the number of redundant broadcasts increases to 4, the resultant additional coverage drops well below 0.05%.

Effect of Contention

Let's say if node X broadcasts a packet and n nodes receive it. If all the n nodes try to rebroadcast the message, then it results in contention since there might be two or more nodes closer to node X which may contend for the wireless medium. It is worth mentioning that when the total number of nodes n is 2 the probability of getting 0 contention free node is about 0.6. Also, the probability of getting at the most 2 contention free nodes turns out to be about 0.4 [10]. Furthermore, as the number of nodes increase to 8, even the probability of getting 1 contention free node decreases rapidly, and the probability of getting 0 contention free node increases to about 80%. This means that we have almost no contention free nodes.

Effect of Collision

The effect of collision is the final part of the Broadcast Problem. The effect of collision is quite serious since in 802.11 MAC one too many behavior is not considered. Tsung.et.al [10] state two main reasons for the occurrence of collision. First, due to the expired Backoff procedure all the nodes within a particular radius might start broadcasting at the same time. For example, consider that a node X broadcasts a packet and its surrounding nodes hear the broadcast transmission. If the surrounding nodes of the node X were previously quite for a long time, then it is quite probable that their Backoff timer could have expired. Thus after the time out of their Backoff procedures and passing their DIFS they may start broadcasting around the same time. Second reason is that, since RTS/CTS forwarding mechanism is not used in broadcast transmission the impact of collision becomes more serious.

2.4.3 AN OVERVIEW OF BROADCAST ALGORITHMS

To overcome the broadcast storm problem, there are many algorithms that are presented in [10], for example Probabilistic Scheme, Counter Based Scheme, Distance-Based Scheme, Location-Based Scheme and Cluster Based Scheme. In probabilistic scheme the rebroadcasting is done with a certain amount of probability, In counter based scheme a counter is initialized to keep track of the number times a broadcast message is received based on a fixed threshold the packets are dropped, In distance based method a relative distance between the hosts is used to make the forwarding decision. The Location Based scheme relies on the GPS (Global Positioning System); if the location of the broadcasting node is known then the additional coverage of the broadcast can be easily estimated and optimized.

The above mentioned methods were proposed as a part of MANET Broadcast Scheme, but in this thesis we are dealing with VANET. Though both are ad-hoc networks, the VANET varies widely from MANET mainly due to its high mobility and finite predictable path of movement. As we know a vehicle may travel at more than 60 miles per hour, at such high mobility the above mentioned broadcast algorithms tend to be largely inefficient in mitigating the broadcast storm problem. Hence in this thesis we consider a widely used algorithm called the Scalable Broadcast Algorithm [11]. The reason for choosing this algorithm has been discussed in section 2.4.4.

2.4.4 CHOOSING THE OPTIMAL BROADCAST ALGORITHM

The Scalable Broadcast Algorithm (SBA) was selected as the second broadcast scheme to implement the group chat application due to its high efficiency. The SBA outperforms most of the Broadcast Protocol as shown in [12]. The paper [12] proposes some valuable results which justifies the use of SBA protocol as a part of this simulation. The following sections describe them in brief.

Delivery Ratio

The Delivery Ratio is defined as the percentage of network nodes which receives any broadcast packet. The delivery ratios for Flooding and Scalable Broadcast Protocol seem to be higher when compared to other Broadcasting Protocols.

Number of Retransmitting Nodes

As we have seen before that as the number of retransmitting nodes increase the area covered by the broadcast packet greatly reduces. It is shown by [12] that the number of retransmitting nodes increases to about 100 nodes for Flooding broadcast if the node density reaches 100 to 110. But whereas SBA Algorithm greatly reduces the number of rebroadcasting node to 20 for a node density of 100, the SBA protocol also outperforms both counter based and probability based schemes. As stated before protocols that minimize the number of redundant transmissions, deliver most packets in congested networks.

End to End Delay

End to End Delay is defined as the time taken for the last node to receive the broadcasted packet. It is stated in [12] that the SBA produces low end to end delay when RAD is adapted as a part of SBA. RAD is defined as Random Assessment Delay which is a timer maintained by a Broadcasting Protocol to keep track of the redundant messages received over a short time interval in order to determine whether to rebroadcast or not. The RAD is chosen between 0 to T_{max} as a uniform distribution. In this thesis the RAD is not implemented instead the native version of the SBA protocol is implemented [11], but even without RAD the SBA protocol has a very low end to end delay.

Effect of Mobility

The protocol used for VANET communication must be very robust to mobility. The SBA performs reasonably well even when the movement of nodes are random [12]; nonetheless, SBA’s performance in high mobility scenario has not been researched. In our simulation scenario we do not have random movement of nodes; however, our scenario involves nodes moving with very high speeds of 60-65 miles/hr. We wanted to see whether the SBA can prove effective even in such environments involving very high mobility.

Transmission Overhead

The SBA is a protocol that takes advantage of its neighbor node’s knowledge (2-hop neighbors) by beaconing. In general the overhead produced by SBA is tad higher that afore mentioned protocols, but optimizing the “Hello” packet interval results is producing a balance between delivery ratio and packet overhead.

2.4.5 SCALABLE BROADCAST ALGORITHM

The Scalable Broadcast Algorithm was proposed in [11], the algorithm uses the local neighbor node information to gain the knowledge about the topology and statistical information of duplicate message to avoid unnecessary rebroadcasts. The Broadcast mechanism retains the goodness of flooding while introducing a little overhead to greatly reduce the number of redundant messages. The main idea behind the algorithm is that a node need not rebroadcast the packet if all of its neighbors have been covered by previous transmissions. In this algorithm a node should have knowledge about its neighbor nodes, which the author terms as “Local Neighborhood Discovery”. In Local Neighborhood Discovery “hello” messages are periodically exchanged between the nodes; the hello messages include the neighbor information, and a node can learn the information about the topology within two hops.

The working of the SBA algorithm is shown in Figure 2.4.1. In SBA algorithm every node maintains a neighbor table which has the information of the node’s reachable neighbors. This information is obtained by periodic exchange of “Hello” packets. To illustrate the working of the SBA algorithm consider 3 nodes S, X and Y. S being the source node, broadcasts the packet and node X receives the packet. This packet contains the information like the source of the packet and the number of nodes covered by this packet (the neighbor list of previous transmittor). Now assume that Y receives the broadcast packet from node X; now, node X would perform the steps shown in Figure 2.4.1. Initially it checks its neighbor table with a newly created table called as the Broadcast Cover Set table. The Broadcast Cover Set or simply Cover Set contains those nodes that are covered by the broadcast of the previous transmitter. If the Neighbour table and the Cover Set exactly matches then there is no rebroadcast by node Y, since the nodes within Y’s range has already been covered by the node X. If the tables does not match then a rebroadcast timer is started this rebroadcast timer is started.

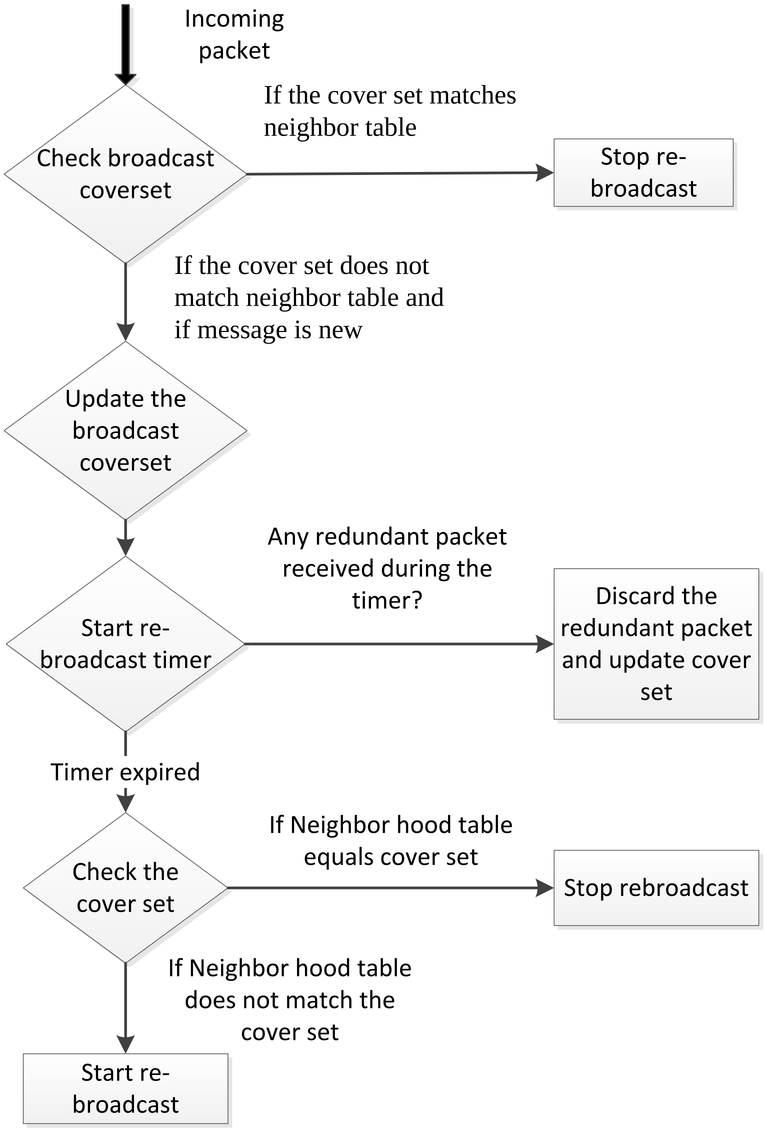


Figure 2.4.1: SBA Algorithm

It is very important to note that the performance of the protocol greatly depends on properly choosing the rebroadcast timer. It is stated in [11] that the best decision would be to let the nodes with more neighbors to rebroadcast earlier, in this way more nodes can be covered in one transmission. The delay time T is give by the equations $T0 = \frac{1+d_m(x)}{1+d(x)}$ and $T = U(\Delta \times T0)$. Where T0 is the minimum rebroadcast timer and T is the maximum broadcast timer, d(x) is the degree of node u, d_m(x) is the maximum degree of neighbors of x, Δ is a small constant delay and U(x) is a function that returns a random number distributed evenly between 0 and x.

During the time frame of the rebroadcast timer, if a redundant packet is received by the node Y, the packet is discarded, but the Broadcast Cover Set is updated based on the neighbor table of the incoming packet. Once the the rebroadcast timer has expired the node Y compares its neighbor table with the updated Cover Set table. If the neighbor table is a subset of the Cover Set table then the rebroadcast is stopped, if not the node Y rebroadcasts the packet.From the SBA's broadcast mechanism we can clearly see that the number of redundant messages are greatly reduced.

2.5 VANET REAL-TIME TRACES

The mobility model developed in this thesis mimics the road layout of I-75S Highway from Detroit to Toledo. The vehicle movement traces were developed using VISSIM; the model parameters were calibrated based on field measurements of the traffic flows on simulated streets. For our experiment, we generated two traces; one, with a low density scenario and other with a high density scenario. The low density layout involves 250veh/hr (vph) travelling at a speed of 65mph (the distance of the layout is not known and the number of injected vehicles are not known). The high density layout depicts the I-75 highway with 1200 veh/hr, travelling at a speed of 65mph (injection rate not known).

The vehicles that participate in the simulation are called as active nodes, so when a node enters the simulation scenario it is marked as an active node. In NS2 it is not possible to allocate and destroy nodes dynamically; hence, an extension has been developed to turn on/off the wireless interface in each node. When a node enters the simulation scenario the wireless interface is turned on and when it leaves, the interface is turned off. As we inject new vehicles into the I-75S trace, the number of vehicles in the environment increases gradually until it reaches a predefined threshold. This stabilization period is known as the warm up phase.

3 RELATED WORK

This chapter explores other works on VANET Comfort Applications. The first section elucidates the purpose of analyzing other works. The final section discusses the research papers which addresses the problems similar to this thesis.

3.1 PURPOSE OF THE STUDY

The need to explore other works related to this thesis is to integrate some valuable ideas and to exploit some of the flaws in previous works. Mentioning the positive and negative aspects would serve as a valuable contribution, aiding further insight in VANET communication. Another purpose of related work study is to clearly distinguish between the results of this thesis with other similar efforts.

3.2 CONCEPTION AND ANALYSIS OF OTHER RESEARCH WORKS

The data and concepts required for this thesis were borrowed from works based on following topics.

- Papers on Modeling Voice Chat
- Papers on VANET infotainment
- Papers on VANET broadcast protocol

The Thesis forms an extension to paper [7]. The concepts that were needed for this thesis can be summarized into the following sub concepts.

Group Chat Behavior

Group chat is one of the most fundamental and most essential part of entertainment. People need social interaction; most of them like to be connected with their friends and family all times. This is the reason for the huge success of social networking websites. Similarly the idea of the thesis was to integrate one of the basic features of social interaction, “Group Voice Chat” in the inter vehicle scenario. Vehicles can form groups to initiate a group conversation while travelling. Paper [7] terms this as *Caravanning*.

As stated in Section 2.3, the first step in designing a real world group chat scenario was to understand the behavior and pattern of a real world group chat. For this reason we went to analyze the distribution that a group talk follows and the most common group sizes for a group talk scenario. Paper [8] thoroughly studies the group talk behavior of people, this study was conducted on those people who play the online multiplayer games which integrates voice chat feature. From this study it shows that the group sizes of 3, 5 and 8 are the most common group while the group size of 5 has the highest frequency with 3 and 8 with a frequency worth for consideration. In this thesis the simulation was performed using the group sizes of 3 and 5. There are other important data that are presented by [8] which includes the maximum talk duration, time etc. but these are not required by this thesis.

To analyze the distribution that the voice chat follows, the work [5] was analyzed to deduce some important data. Firstly we found that the voice chat algorithm follows the Weibull distribution. The Voice chat is modeled using two parameters; one is the Talkspurt and the other being the Silence Duration. The Talkspurt has a value of ʎ = 7.6, k = 2.6 as the Weibull distribution parameters and the Silence has the value of ʎ = 58.0, k = 1.7. It is also important to note that none of the papers bring out the idea of integrating voice chat to vehicular infotainment, though there are many papers that talk about individually modeling voice chat application.

Infotainment in VANET

Infotainment in VANET has been a recent attraction to the Vehicular Community. A recent research work [13] talks about the feasibility of Multiplayer Games over Vehicular Ad Hoc Networks. The author explores the QOS metrics that should satisfy the Application. The author in [5] proposes a similar concept called as Road Speak but the paper does not implement the Road Speak Application over VANET technology, instead it uses 3G cellular technology for its validation. As described in the above section some of the Roadspeak parameters related to the Weibull Distribution have been borrowed from paper [5].

VANET Broadcast

There are many work that target broadcasting in MANET (mobile ad hoc networks), however there are only few works that discuss about broadcasting in VANET. Though most of the MANET Broadcast protocols are directly used for VANET there some major differences which demands modification of MANET broadcast to make it feasible for inter vehicle communication. The first difference is the mobility of vehicles. Firstly Vehicles travel at greater speeds than MANET nodes, secondly the fixed pattern movement of vehicles.

Though there are many broadcast algorithms proposed Flooding has been one the most basic broadcast algorithms that perform reasonably well for both VANET and MANET scenarios. The original flooding broadcast protocol was proposed in [9], the flooding broadcast in this thesis has been implemented strictly according to this native paper. Though flooding is a suitable broadcast mechanism for our scenario there are few disadvantages with flooding, which are explained in section 2.4.3.The author of [12] also gives a detail review of other broadcast protocols and measures their performance. The briefing of these protocols was discussed in the section 2.4.4.

The other protocol that has been used to test the Voice Chat algorithm is the SBA (scalable broadcast algorithm), as proposed by [11]. The SBA performs very well when compared to other broadcast algorithms, though the overhead is a bit higher for SBA. It outperforms other broadcast algorithms in content delivered percentage, end to end delay and number of redundant packets. The working of SBA was described in section 2.4.5. The NS2 version of SBA algorithm was borrowed from [6].

4 IMPLEMENTATION OF SIMULATION

This section describes the implementation details of our simulation framework. Section 4.1 gives an overview of VANET simulation; section 4.2 explains the implementation of voice chat algorithm, and section 4.3 describes the implementation of broadcast algorithm in NS2.

4.1 SIMULATION ARCHITECTURE

The simulation methodology is shown in Figure 4.1. The mobility trace file is generated using the VISSIM microscopic simulator. The VANET TCL script requires the mobility trace as input, and this is generated by the VISSIM microscopic simulator [1].

The TCL VANET script also requires the wireless radio control, and the voice chat groups as input parameters. The wireless radio control is used to turn ON the radios of those vehicles that enter into the simulation scenario and turn off the radio for those vehicles that leave the simulation scenario. The voice chat group file sets the vehicle groups and the size of the group that involve in the voice chat. In this simulation, we set two different group sizes and generate 10 different groups for each group size.

The NS2 (with VANET extension) interfaces with the TCL code to execute a simulation that closely matches a real world voice chat between moving vehicles. The final output from the simulation is then parsed to obtain the required results.

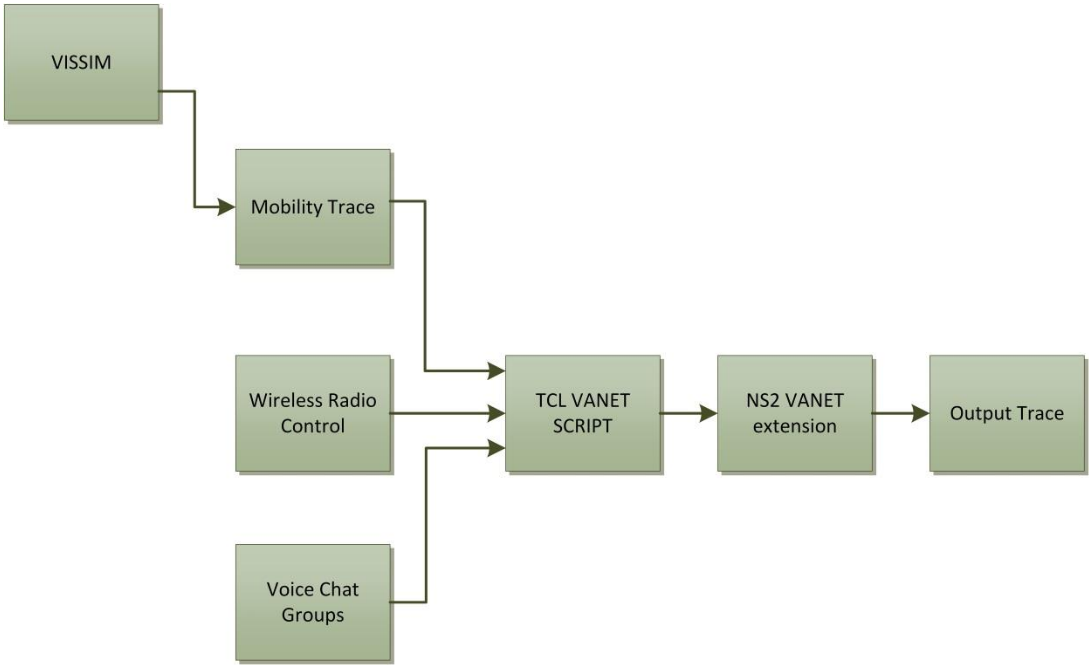


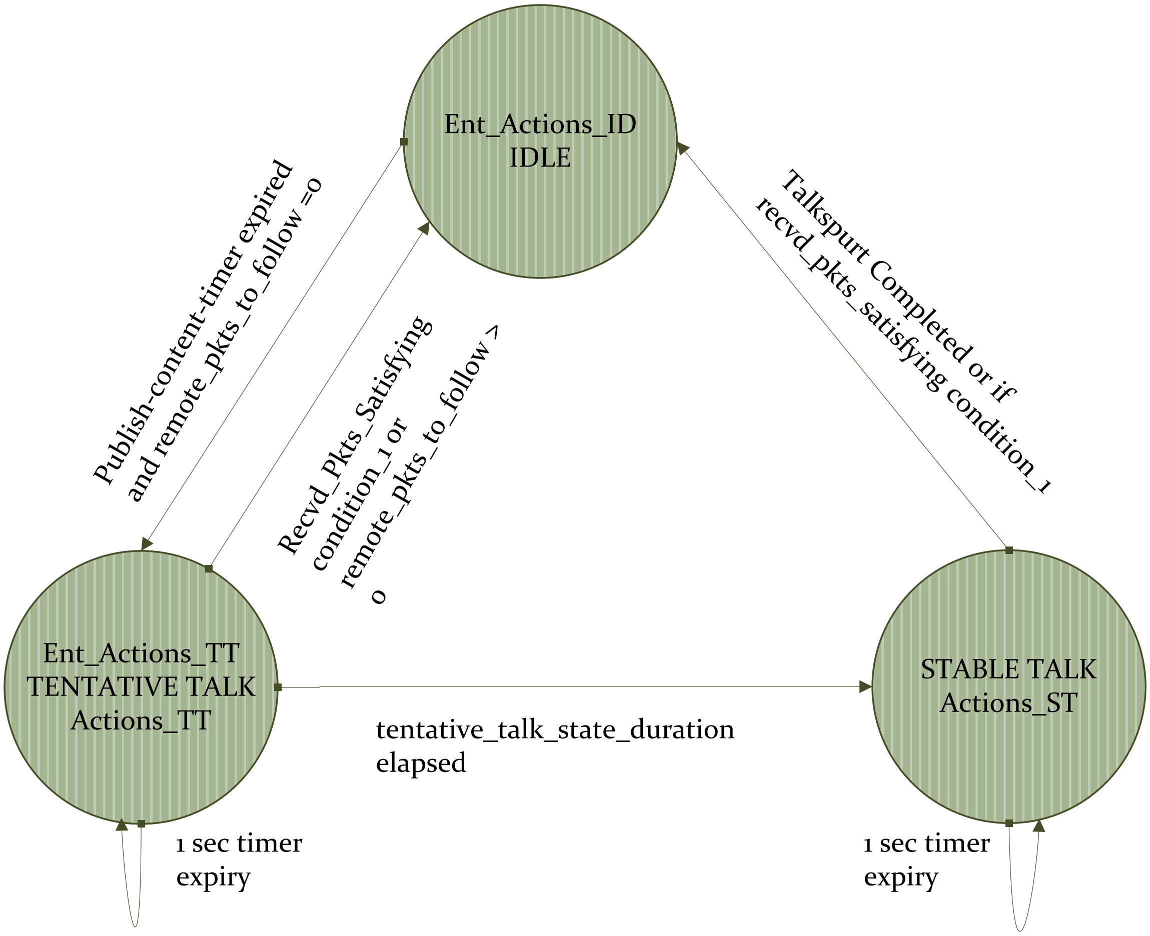
Figure 4.1: Simulation Overview

4.2 VOICE CHAT ALGORITHM

There are three states for the algorithm, which is shown in Figure 4.1. The *Idle state* is when none of the nodes communicate. The *tentative talk* state is a transition state, which is used for contention resolution; the *Stable Talk* state is where a node starts its communication with its group members. We present the description of variables that is used in our algorithm in Table 4.2.

A node goes from *idle state* to *Tentative Talk* state when its *publish-content-timer* expires and when the *remote_pkts_to_follow* becomes zero. The *publish-content-timer* follows a Weibull silence period as described in Section 2. The *remote_pkts_to_follow* is stored in every node's table, this is essential for a node to keep track of the number of packets the publishing node has to send. The *tentative talk* duration is a very short period which is about 10% of the talkspurt length. If a node begins publishing while the other node is publishing, depending upon the *remote_pkts_to_follow* value the node may be sent back to *idle state* or may continue publishing. If the *remote_pkts_to_follow* is not equal to zero this means some other node has already been publishing the packets. On the other hand there are instances where a node's *remote_pkts_to_follow* value may be outdated. Such a situation is shown in Figure 4.3. We can see that node *V1* sends the *remote_pkts_to_follow* count to be 0, which signifies that node *V1* has finished talking, but node *V5* does not receive this count. *V5* enters the tentative talk mode by checking whether the count of *remote_pkts_to_follow* is zero or not. Since *V5* does not receive the previous packet from node *V*, it has an outdated count value. Hence, *V5* goes back to idle state without starting the communication.

To avoid this problem a *remote_pkt_time_stmp* variable is maintained. This variable denotes the time at which the remote packet count was received. If this value is lesser than the current time plus a fixed threshold value then the *remote_pkt_time_stmp* variable is reset to zero. A node which has been previously talking can be interrupted by the current node that has entered into the *tentative talk* state if it has a fairness value greater than the previously publishing node.



condition_1 = (((incoming_speaker_fairness greater than local_fairness_value)or (incoming_fairness_value == local_fairness_value) and (incoming_node_id > my_node_id)) and (current_state == TENTATIVE_TALK))

Figure 4.2: Voice Chat Algorithm

If the fairness values are same, then the decision is made based on the node ID. The node with greater node ID would preempt the other node from talking. *Stable Talk* state is the state where the actual communication between the vehicles takes place. A node spends most of its talking period in the *Stable Talk* state. The interruption of the node follows the same logic as that of *Tentative Talk* state.

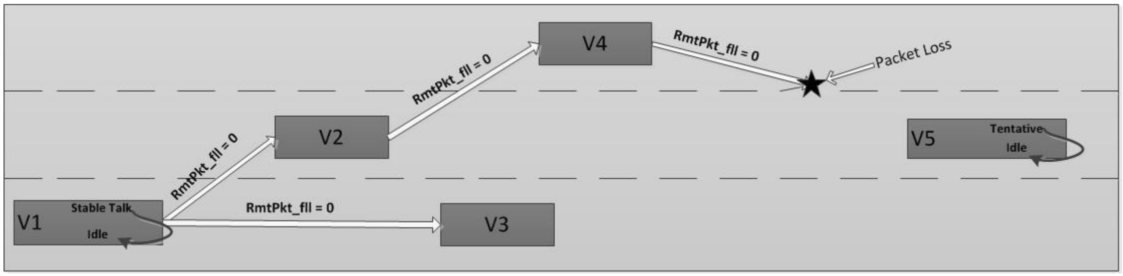


Figure 4.3: A scenario describing the effect of outdated remote packets to follow (remote_pkts_to_follow) count

The variable $talkspurt\_length$ decides time for which a vehicle talks. The $talkspurt\_length$ variable is set to a Weibull talk spurt value as explained in Section 2. When a node completes its talk duration without interruption it again goes back to the idle state where its Weibull silence duration is set. This Algorithm is repeated for the entire simulation duration.

4.3 VOICE CHAT EXTENSION FOR NS2

The voice chat algorithm is implemented over the NS2 VANET module [3]. The algorithm stated in section 4.2 is being implemented over the developed NS2 VANET extension [7]. The Publish Data module which is explained in Section 4.3 has been modified to include the Voice Chat Application code so that the communication in VANET mirrors a Realistic Voice Chat behavior. The Send and Receive modules of the VANET code has been modified to include the Application. The variables used in this algorithm are shown in Table 4.3. The Algorithm 4.3 shows the implementation code.

Table 4.3: Group Chat Parameters

Symbol	Meaning
Talkspurt_length	a weibull random variable determines the talk duration for a speaker ($\lambda = 7.6$, $k = 2.6$)
Silence_interval	a weibull random variable determines the silence duration or inter-packet interval ($\lambda = 58.0$, $k = 1.7$)
Tentative_talk_state_duration	A speaker is in the TENTATIVE_TALK state for at most this length of time. During this time, a more aggressive speaker can preempt the first speaker.
Tot_talkspurt_pkt_count	This value is obtained as follows: Tot_talkspurt_pkt_count = talkspurt_length * 12kbps/1kb (assuming 12kbps is the compressed voice data rate and the 1kb packet size at the MAC layer)
Talkspurt_pkts_to_follow	This value is put into the published content packet and indicates the number of packets that will follow for this talkspurt. Talkspurt_pkts_to_follow = Tot_talkspurt_pkt_count – talkspurt_pkts_tx
Speaker_fairness_value	Varies from 1 to 20. Increases in steps of 2. Higher the value, higher the claim to speak.
remote_pkt_time_stamp	Varies from 1 to 20. Increases in steps of 2. Higher the value, higher the claim to speak.
talk_status	Set to 1 if total packet count for a single talkspurt is completely published by a node. Set to 0 if there are remaining packets to be sent.

Algorithm 4.3

```
1)  Ent_Actions_ID:
If (state == IDLE)
{
          Derive silence-interval as a weibull random variable
          Publish-content-timer = silence-interval
}

 2)  Ent_Actions_TT:
          If(remote_pkts_to_follow != 0 && (current_time – remote_pkt_time_stamp >
          CONSTANT_VALUE)
          {
             remote_pkt_to_follow = 0
          }
          If (previous_state == IDLE) && (remote_pkts_to_follow > 0) {
              Decrement speaker_fairness_value by two steps
              Move to IDLE state
           } else if (pkt_to_follow == 0 && talk_status != TALK_SPURT_SENT) {
             Derive talkspurt-length as a weibull random variable
             Initialize pkts_to_follow to Tot_talkspurt_pkt_count. (see table for definition) talkspurt_start =
          current-time
             state = TENTATIVE_TALK;
          }

3)  Actions_TT:

          if (state != TENTATIVE-TALK) return
          if (current-time - talkspurt_start < tentative_talk_state_duration) {
             start TT_1sec_timer
            send the next 1kb packet
            decrement pkts_to_follow
```

Algorithm 4.3 Continued

```
        }
      if (pkts_to_follow == 0)
      {
        talk_status = TALK_SPURT_SENT
      }
      else {
        Move to STABLE_TALK state
      }
4)  Actions_ST:

      if (current-time – (talkspurt_start + tentative_talk_duration) < talkspurt-length && pkts_to_follow
      > 0)  {
            send the next 1kb packet
            decrement pkts_to_follow
      if (pkts_to_follow == 0)
      {
        talk_status = TALK_SPURT_SENT
      }
      start ST_1sec_timer
      } else {
            Increment speaker_fairness_value by 2
              If (pkts_to_follow != 0) pkts_to_follow = 0;
            Move to IDLE state (i.e., state = IDLE);
            talk_status = TALK_SPURT_RESET;
            Send the Last 1kb packet
      }

 5) Receiving a packet
    If (state == TENTATIVE_TALK || state == STABLE_TALK) {
        If (remote_fairness_value < speaker_fairness_value) {
                Move to IDLE state (i.e., state = IDLE)
                decrement speaker_fairness_value
                pkts_to_follow = 0
        }
 } else if (state == IDLE) {
 remote_pkts_to_follow = pkts_to_follow from the received packet
 remote_pkt_time_stmp = remote_pkt_time_stmp of the received packet
 }
```

4.4 IMPLEMENTATION OF BROADCAST ALGORITHM

This thesis compares the performance of voice chat application using two broadcast algorithms. First, we use the Flood broadcast scheme, which was explained in Section 2.4.1. This algorithm was implemented by Rao.et.al [7], and evaluated in their paper. In this Thesis, we extend their simulation module by incorporating the new voice chat module, and a new SBA broadcast module. A modified version of the SBA algorithm called the BCAST was implemented by Kunz [6]. We used the same code by modifying and adding new methods to change the code back to the original SBA algorithm.

The working of SBA was described in Section 2.4.5. The BCAST [6] was implemented by altering and tweaking some parameters in the original SBA broadcast algorithm [11]. The author implemented a NACK mechanism which was not a part of the native SBA algorithm. In this thesis, we altered the BCAST code to follow the native SBA version. Other necessary additions in the BCAST code were performed to suite the NS2 VANET code. In the following paragraphs, we explain the some of the key modules in the NS2 version of the SBA broadcast algorithm.

The SBA is basically divided into two classes, namely the *local neighborhood discovery* and the *data* broadcasting. The local neighborhood discovery involves periodic broadcast of *Hello* packets, which enable the node to know its two hop neighbors. The data broadcasting class has all the modules involving the broadcast of the packet, forming neighborhood table and forming the cover set table. The *RetransmitTimer* class is used for deciding and varying the retransmit time; the *start_rtx_timer* module starts the retransmit timer, saves the copy of the received packet and discards the packet based on the retransmit time; the *cancel_rtx_timer* is used to cancel all the retransmit time that was set. The module *recvSBA* saves the time at which the packet was received; it also piggy backs the *Hello* message. The *SBA_forward* module keeps track of the neighbors that were not covered by the broadcast by comparing the neighbor table and the cover set table. It finds the neighborhood information for the sender and processes this against the information stored in each broadcast. The module also copies the received packet for further retransmission and eliminates all neighbors that are covered by the previous sender. The *SBA_forward* module also enqueues the packet if there are uncovered neighbors and determines the right amount of jitter. The *sendPacket* module is a part of the SBA class which adds the neighbor table into the packet; it cancels and reschedules the packet. It also stores the information about the packet that is being sent and sends the packet. Dealing with retransmission requests and information about NACK packet has been disabled for this simulation since we are using the original SBA broadcast. The *sendHello* keeps info about the *Hello* being sent by the node and fills in the neighborhood info. The *nb_delete* module is called when we receive explicit notification that a neighbor is no longer available. Finally the *nb_delete* module purges all timed-out neighborhood entries and runs the *Hello* interval after every 1.5 seconds.

5 EXPERIMENTAL EVALUATION

In this section, we detail the results of our NS2 simulation that was executed over real time VANET traces generated by VISSIM [1].We performed exhaustive set of experiments involving group sizes of 3 and 5. The simulation time was set to 1800 seconds, and the experiments were repeated for 10 randomly chosen vehicle groups. Throughout the results section, we have compared the performance of the FLOOD broadcast protocol with the SBA protocol using two different node densities. First, a low density scenario involving 250vph (vehicles per hour) was chosen to run our experiments; second, a high density scenario comprising 1200vph was chosen to compare the effects of node density on the voice chat algorithm. Surprisingly we found that FLOOD broadcast is more suitable for our scenario than the SBA broadcast. We show the results of our experiment in the following sections.

5.1 CONTENT DELIVERY FOR FLOODING BROADCAST

The total number of packets delivered is one of the key factors in determining the feasibility of the voice chat application. The content delivery ratio measures the number of voice packets that are received by the group members involved in the group chat scenario. In this section, we show the results of the content delivery for FLOOD broadcast protocol with a vehicle density of 250vph. Figure 5.1 shows the average content delivered for a group size of 5 vehicles. We can observe a reasonable difference in the percentage of content delivered for hop counts 1, 2 and 3, with a range of 200m (Figure 5.1). For hop count 1 we see only 68% of the content being delivered, this is because if the destination node is not reachable within one hop then the packet would be dropped and not forwarded. Hop counts 2 and 3 have better percentage of content delivery than hop count 1 since the number of reachable set increases with the increase in hop count. Reachable set is defined as the number of nodes that receive the broadcasted packet. As proposed by Rao et.al [7] there is a steep increase in the reachable set with the increase in transmission range and hop count. This is because, as we increase the transmission range and hop count, the area covered with each extra hop is larger.

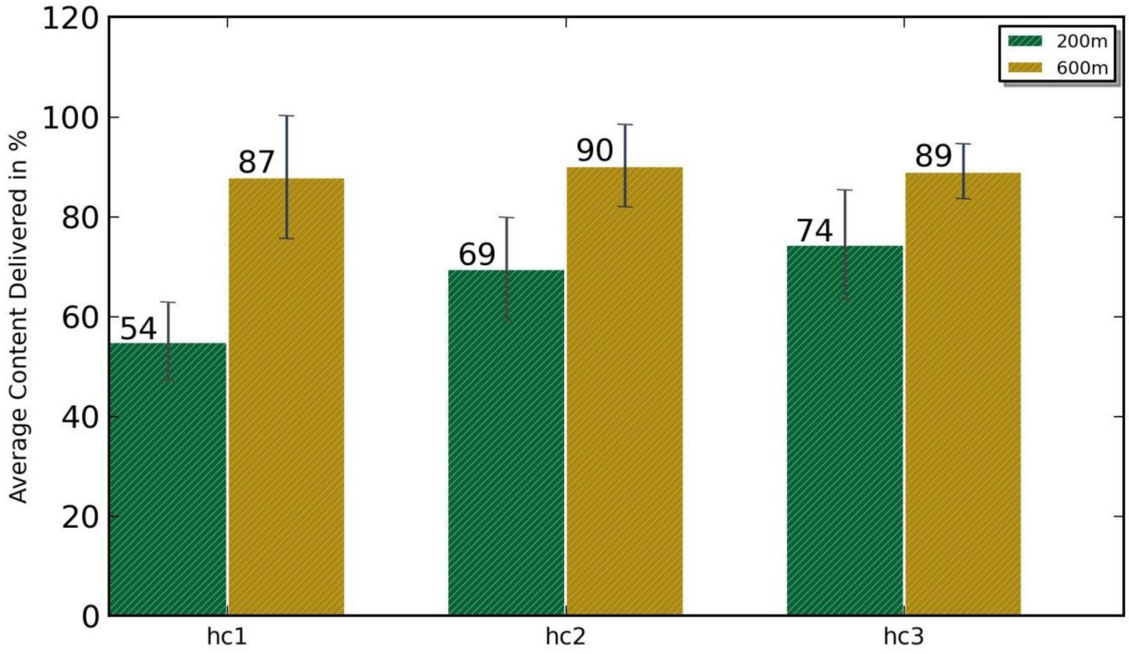


Figure 5.1: Average Content Delivered Group Size 5

For a range of 600m there is a distinct increase in the content received percentage, but as we increase the number of hops there is no significant increase in the output. One of the main reasons for this trend is that the reachable set does not increase drastically with the increase in number of hops when the transmission range is set to 600m. Since the transmission range is quite large, the destination nodes are reached with fewer hop counts; hence, increasing the hop count does not increase the content received. Another important factor for this phenomenon is the vehicle density. Figure 5.1 presents the result of a low density scenario (250vph); lower density implies more packet loss since a node might not find any other node within its range to forward the packet. Figures 5.2 and 5.3 show the CDF of the content delivered.

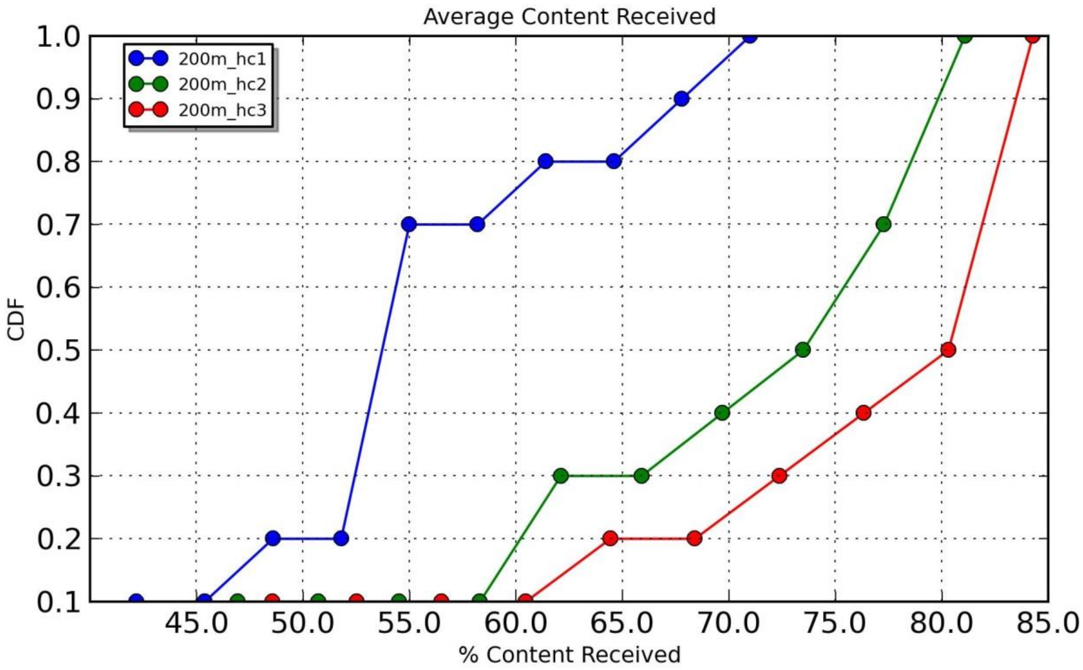


Figure 5.2: CDF for Average Content Delivered for Group Size 5 and Range 200m

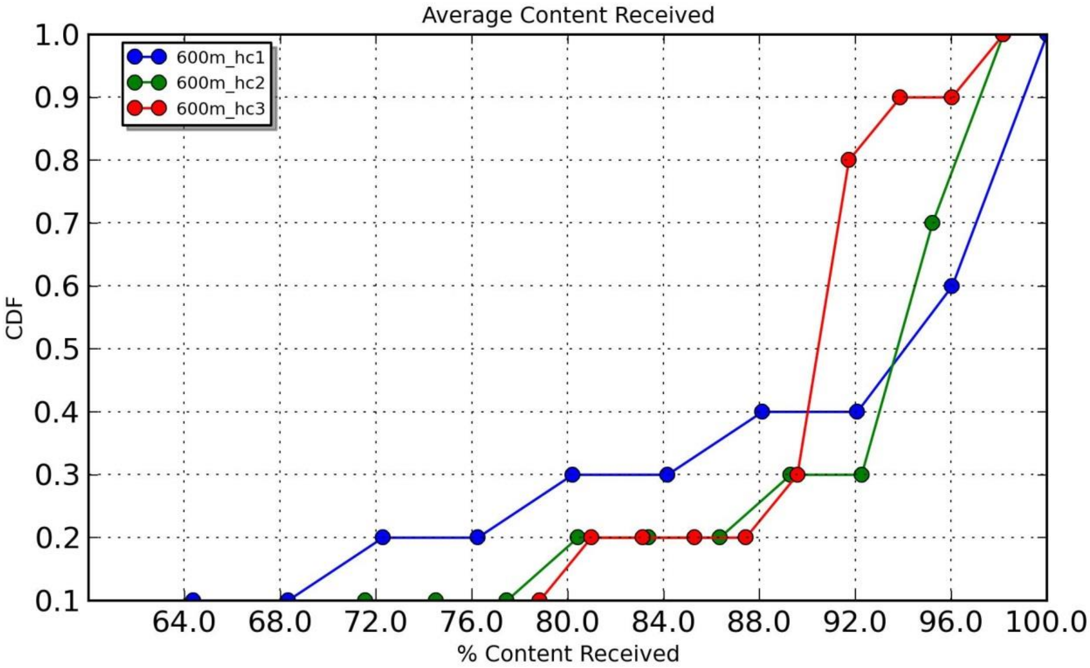


Figure 5.3: CDF for Average Content Delivered for Group Size 5 and Range 600m

From the Figure 5.2, we can observe a greater variation in the content delivered for hop count 1. The percentage of content delivered is concentrated between 46% and 100%. However, for hop counts 2 and 3 the values range from 60% to 100%. This variation gets narrow for 600m; Figure 5.1.3 illustrates this phenomenon in which the content delivered is concentrated between 70% to a 100% for all 3 hop counts.

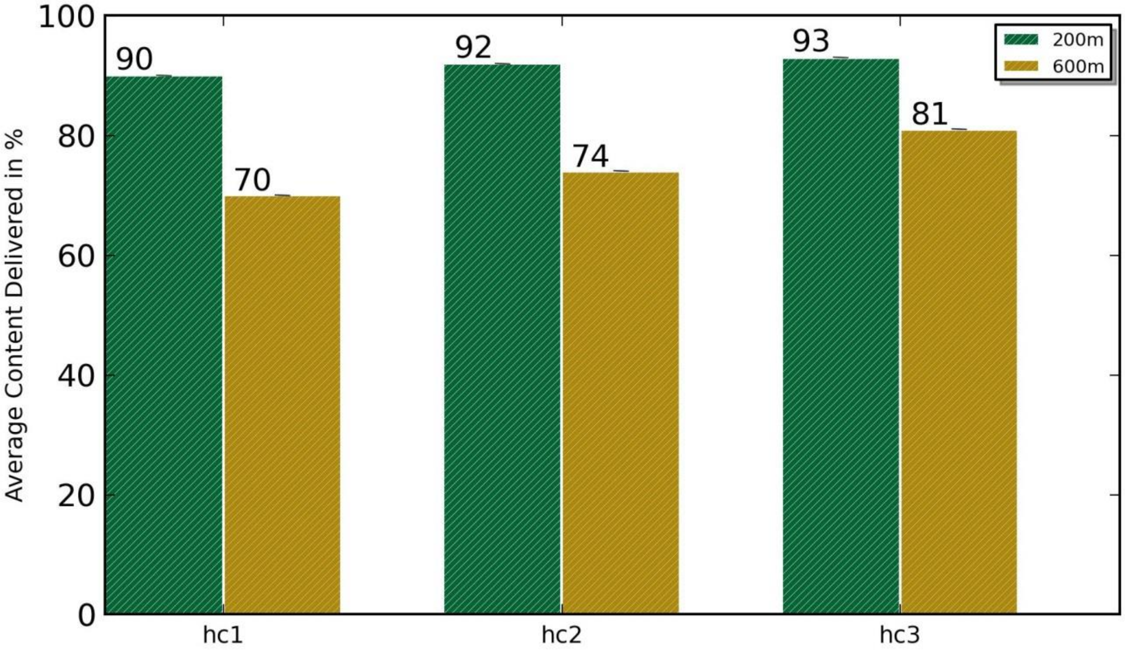


Figure 5.4: Average Content Delivered Group Size 3

Experiments were also performed to analyze the impact of group size on the content delivery. We observe a better content delivery for 200m and 600m ranges for a group size of 3 (Figure 5.4). The reason for the increased content delivery ratio is due to the lower distance of separation between the group members. When the group size increases, there is a greater chance of a vehicle getting separated from the rest of its group members and this is much more severe in lower hop count scenario.

5.2 CONTENT DELIVERY- FLOOD VERSUS SBA

This section compares the content delivered percentage between Flood and SBA broadcast scheme on our voice chat algorithm. As discussed in section 2.4.5, setting of rebroadcast timer for SBA is quite critical in achieving higher content delivery. After many experiments we found that a rebroadcast time of 0.05sec is optimal for our scenario. Figure 5.5 shows the comparison of using the broadcast protocols.

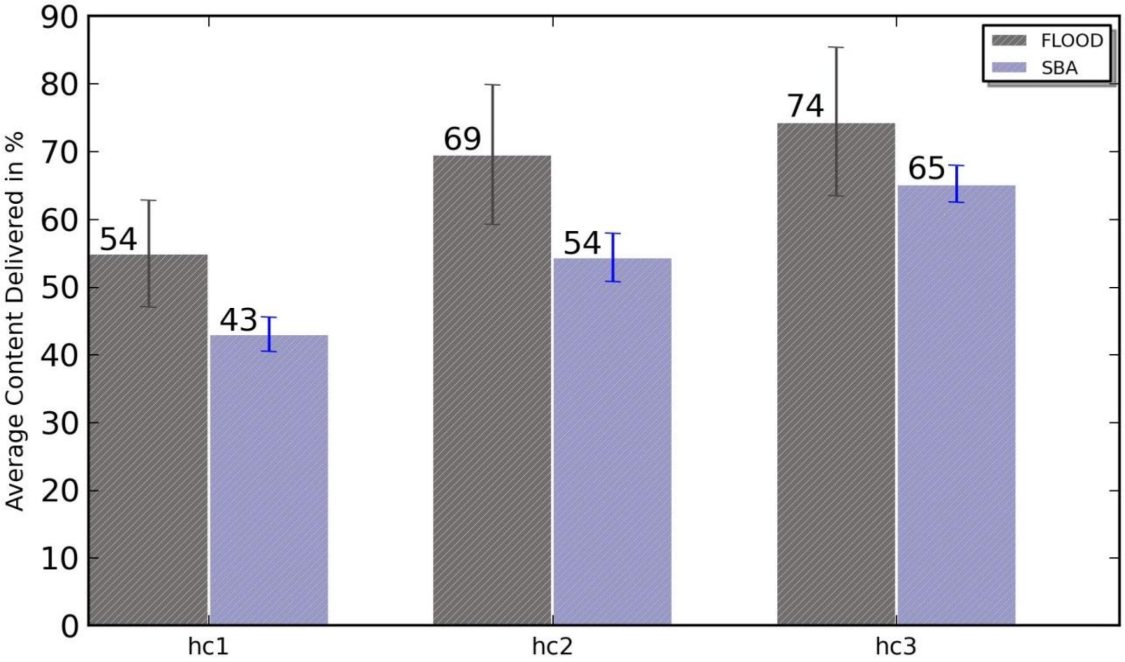


Figure 5.5 Average Content Delivered - Flood versus SBA Broadcast with 200m range

We observe a lower content delivery ratio for SBA broadcast when compared to Flood broadcast. The SBA falls behind the Flood by approximately 10% in all the hop counts. The Flood broadcast produces a content delivered percentage of 69 and 74% for hop counts 2 and 3 respectively, while the SBA algorithm results in a decrease of 15% in hop count 2 and 13% in hop count 3 over the Flood algorithm. This observation shows that the SBA algorithm might not be extremely efficient in high speed VANET environment.

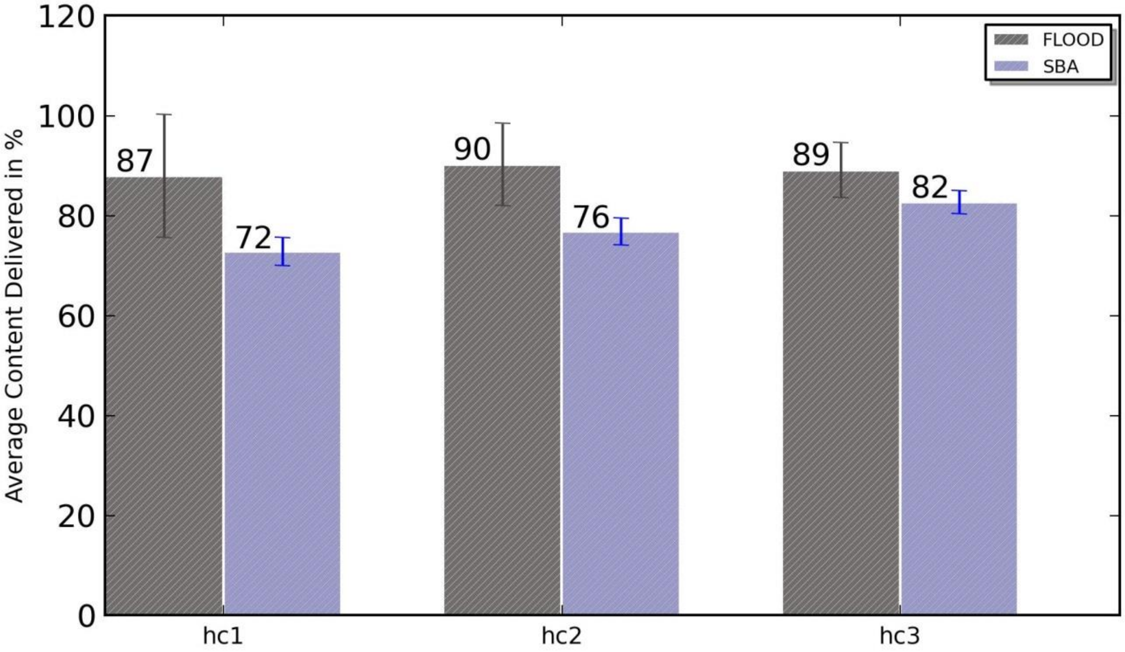


Figure 5.6 Average Content Delivered – Flood versus SBA Broadcast with 600m range

The same experiment was performed by increasing the range from 200 to 600m. Increasing the transmission rage has definitely resulted in an increased content delivery; however, we still notice a significant difference between Flood and SBA's content delivery ratio. This trend is similar to that of 200m range in which the SBA produced lower content delivery. Figure 5.7 and 5.8 shows the result of the same experiment for a group size of 3 vehicles. It is quite clear that lower group sizes results in higher percentage of content delivery for both Flood and SBA. Nonetheless, it doesn't result in improving the performance of SBA broadcast algorithm. In all scenarios the SBA clearly lags behind the Flood broadcast.

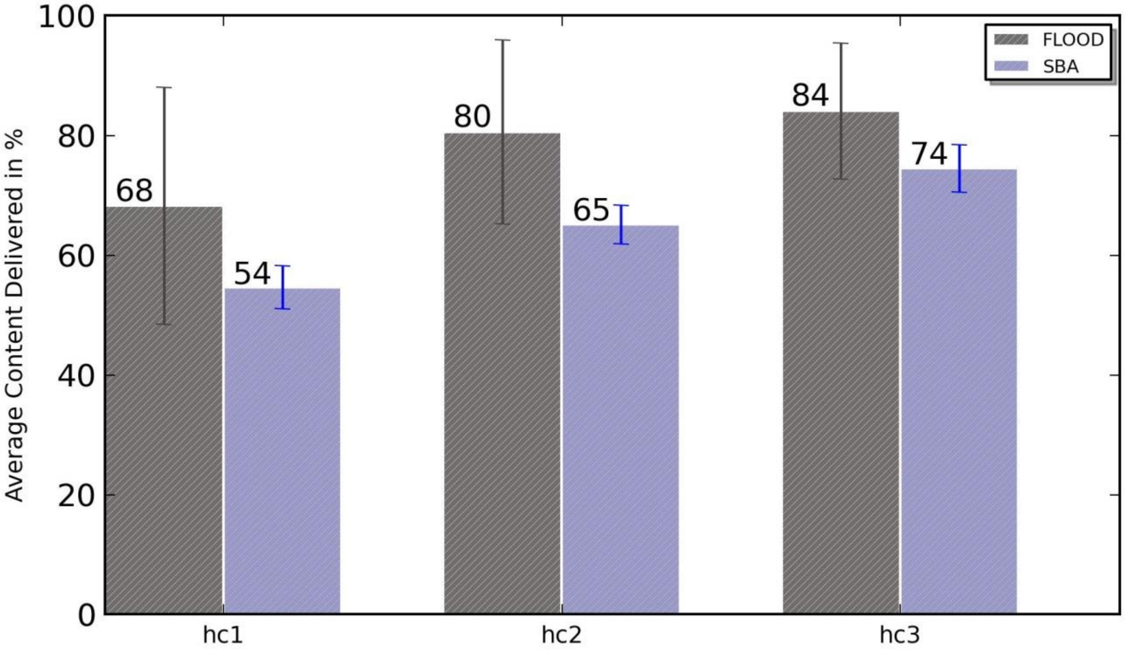


Figure 5.7 Average Content Delivered – Flood versus SBA Broadcast group size 3 with 200m range

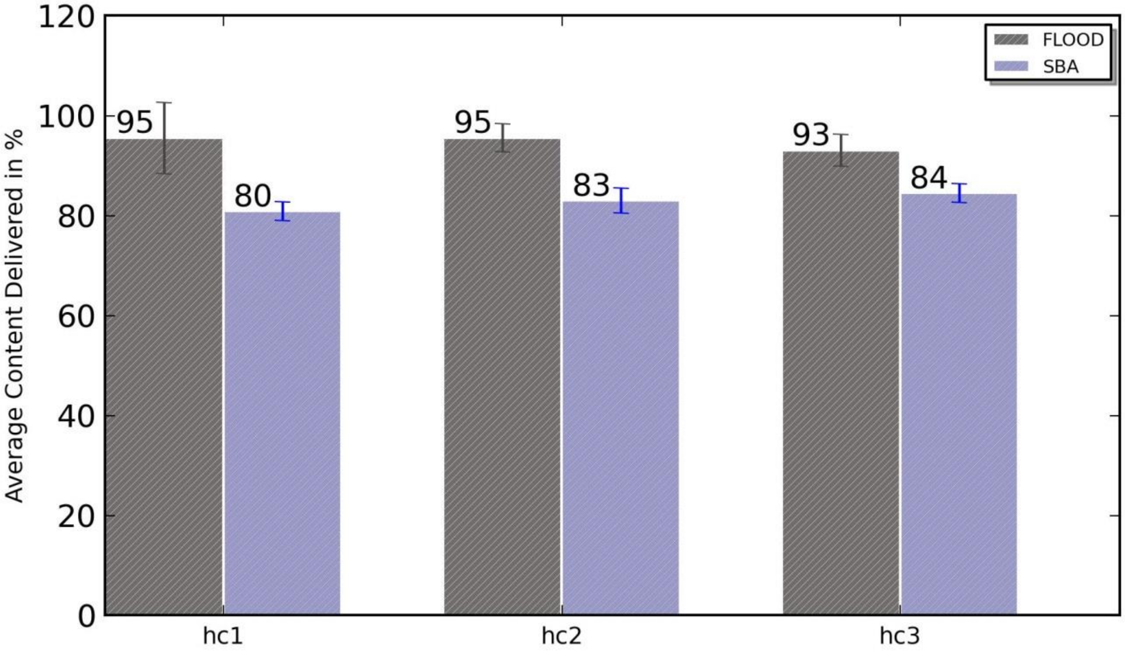


Figure 5.8 Average Content Delivered – Flood versus SBA Broadcast group size 3 with 600m range

5.3 EFFECT OF VEHICLE DENSITY OVER CONTENT DELIVERY

We increased the vehicle density from 250vph to 1200vph to study the performance of the two broadcast protocols. We witnessed an increase in the content delivered percentage with the increase of node density. This improvement is distinctly notable at lower hop counts since higher node densities result in better forwarding of packets and tighter connectivity between the group members. The improvement in content delivery becomes slimmer when the transmission range in increased from 200m to 600m, and it remains slim through all the hop counts; figure 5.11 shows this observation. The same set of experiments was repeated for a group size of 3 nodes. The results are presented in Figures 5.12 and 5.13.

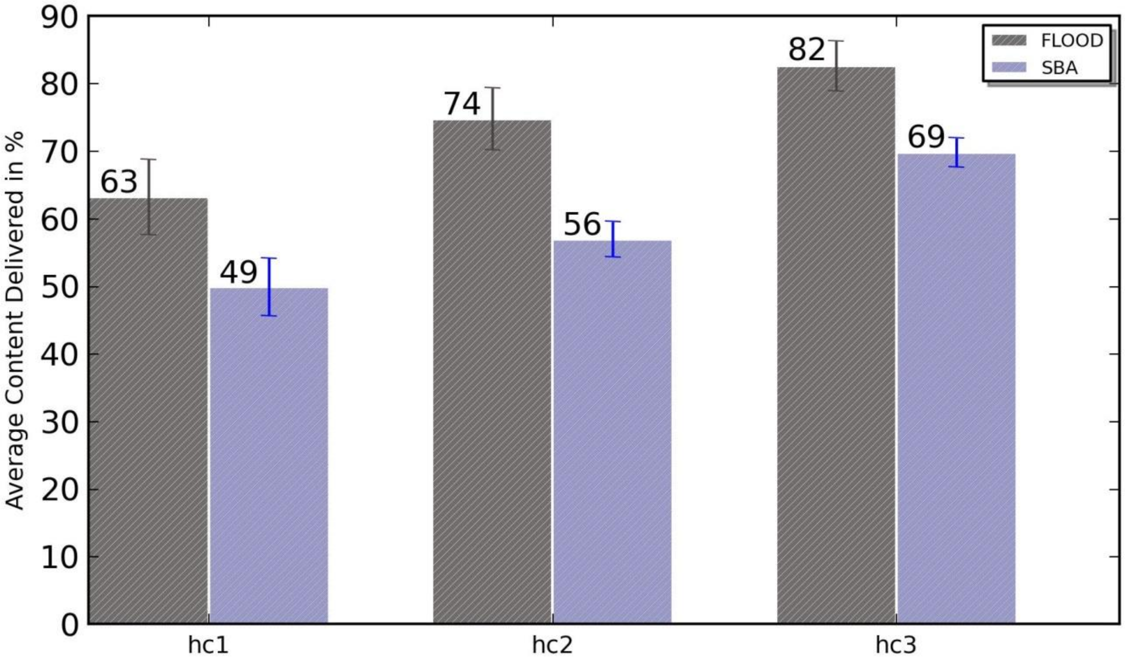


Figure 5.10 Average Content Delivered – Flood versus SBA Broadcast group size 5, 1200veh/hr with 200m range

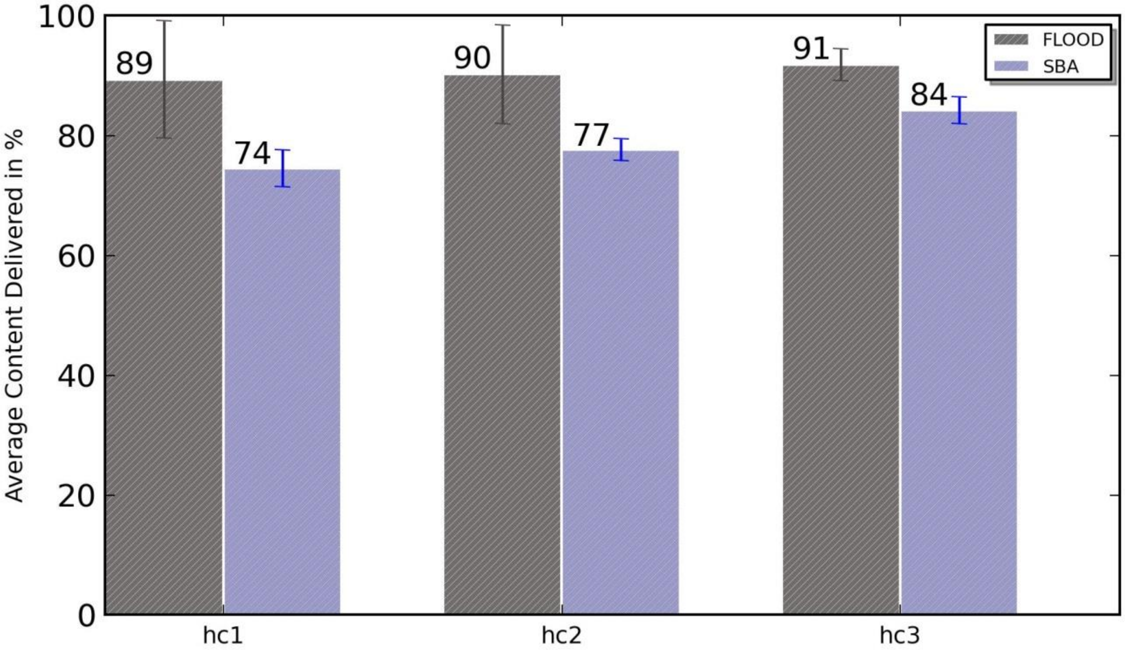


Figure 5.11 Average Content Delivered – Flood versus SBA Broadcast group size 5, 1200veh/hr with 600m range

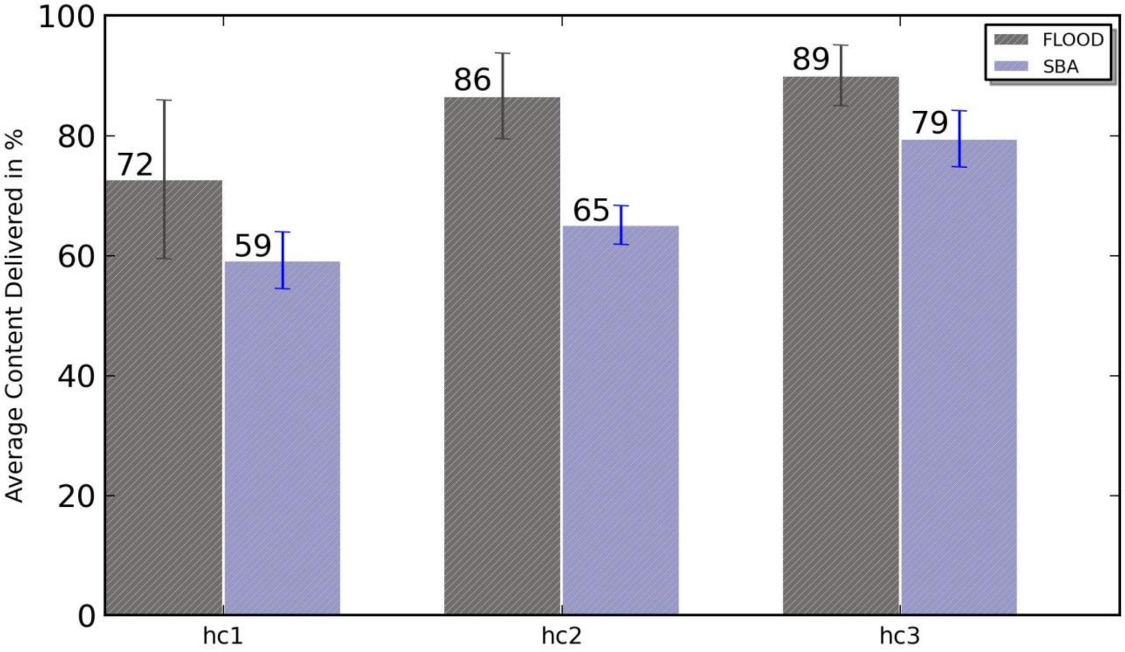


Figure 5.12 Average Content Delivered – Flood versus SBA Broadcast group size 3 , 1200veh/hr with 200m range

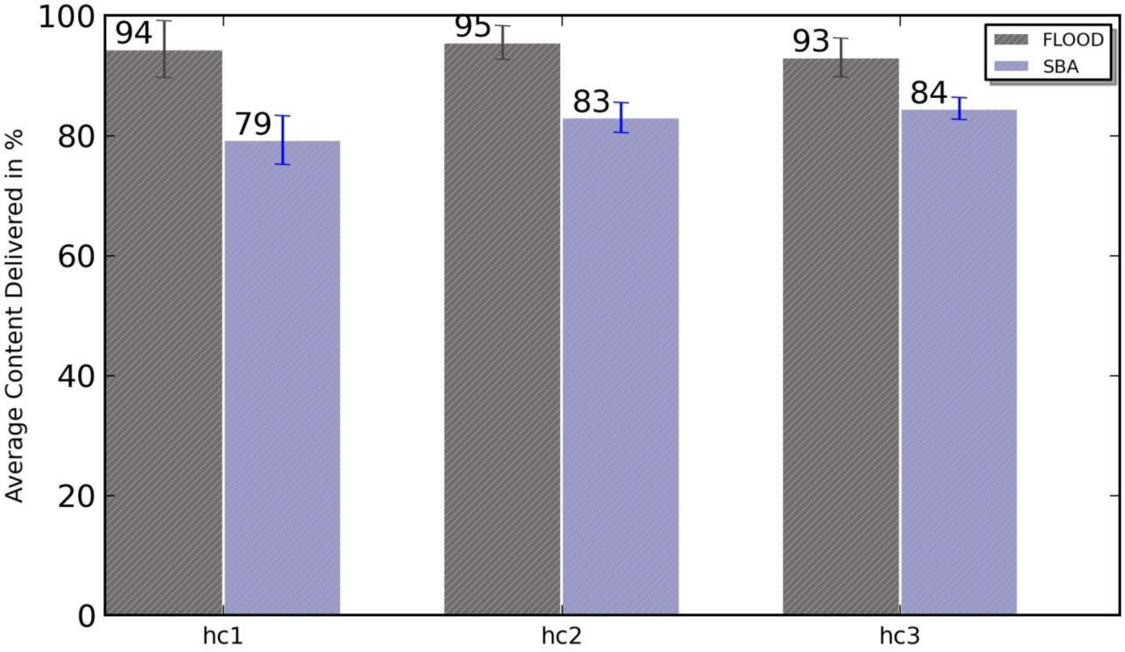


Figure 5.13 Average Content Delivered – Flood versus SBA Broadcast group size 3, 1200veh/hr with 600m range

6 HOP COUNT DELAY

Hop count delay is defined as the time taken for the broadcasted packet to reach its respective group members. In this section, we measure the hop count delays of Flood and SBA broadcast schemes that implement our voice chat algorithm. Individual delays for each group member are calculated for the entire talk duration. For every packet that is being broadcasted by a group member (node), we calculate the time taken to reach the group members. We repeat this process for every node that belongs to the group involved in voice chat scenario. We finally take the average of all such delays and plot them for hop counts one, two and three. In section 6.1, we present the hop count delay statistics for Flood broadcast algorithm; in section 6.2 we compare the hop count delays of Flood and SBA broadcast schemes, and in section 6.3 we study the effect of node density on hop count delays.

6.1 DELAY STATISTICS FOR FLOOD BROADCAST ALGORITHM

From Figure 6.1 we observe an increase in the delay value for hop counts 2 and 3 for a group size of 5 vehicles.

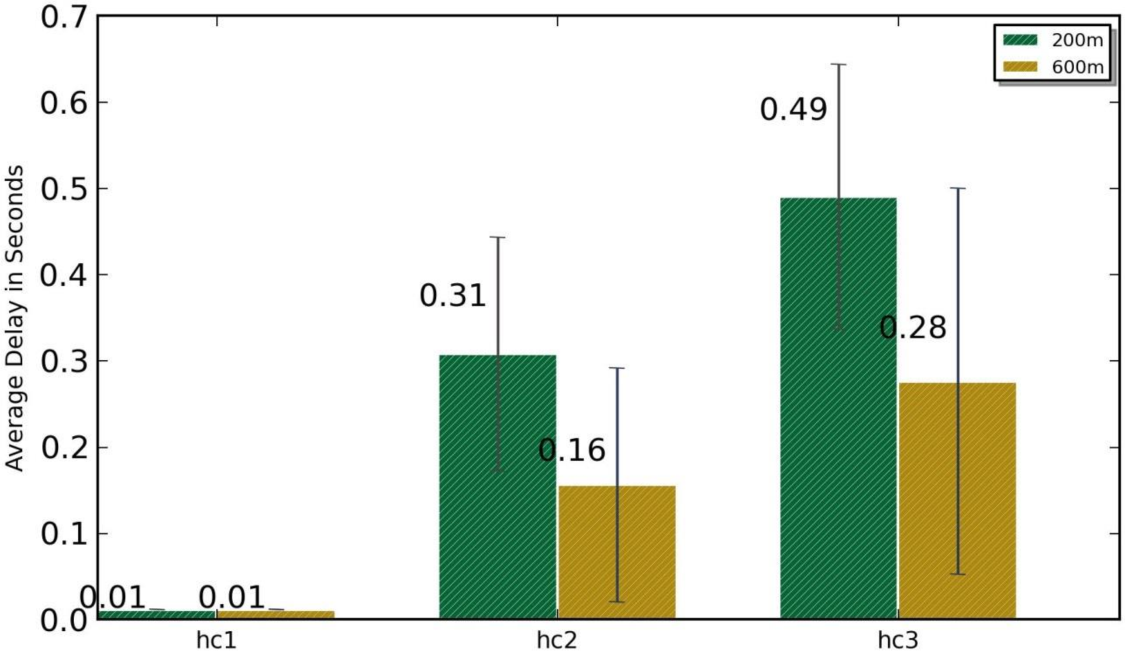


Figure 6.1: Average Delay Group Size 5

The standard deviation (The error bars on the graphs) is quite large for hop counts 2 and 3 irrespective of the transmission range. This can be attributed to the distance of separation between the vehicles. For a group size of 5, most groups have more than 1600m as the maximum distance of separation. The distance of separation need not be uniform for all the group members. Some vehicles tend to travel closer maintaining a distance of 100-200m while some maintain a larger distance of separation from the group members. The traffic conditions of the road play an important role in determining how detached or attached the vehicles of the group are. We can observe from the Figure 6.1 that the standard deviation for the range of 200m, hop count 2 to be 0.15sec; the groups having such small delays are those which travel closer, and have a maximum distance of separation between 150m to 200m.

The impact of increasing hop counts on the delay is quite clear from the Figure 6.1; we observe an increase in the delay with the increase of hop counts. For both 200m and 600m we observe an increase of 0.15sec as we move from hop count 2 to hop count 3. Since there is a delay associated with every hop count, the scenarios where the packet takes 3 hop counts to reach the destination result in higher delays. The significance of the transmission range is also quite visible from the delay graph, as we increase the transmission range from 200m to 600m the delay value drastically reduces. When the range is set to 600m, we observe a delay of 0.16sec for hop count 2 and 0.28 for hop count 3, compared to 0.31 and 0.49seconds for 200m.

We further justify the reason for high standard deviations in hop count delays by plotting the CDF plots of average delays. Figure 6.2 shows CDF values as low as 0.1sec for hop count 2 and 0.3 seconds for hop count 3, and delays as high as 0.7 and 0.8 seconds for hop counts 2 and 3 respectively. Delays greater than 0.8 seconds are observed if the separation distance of any one of the group members increases beyond 1600m. Figure 6.3 shows the similar CDF plot for a range of 600m; we observe a lower delay due to the increase in transmission range.

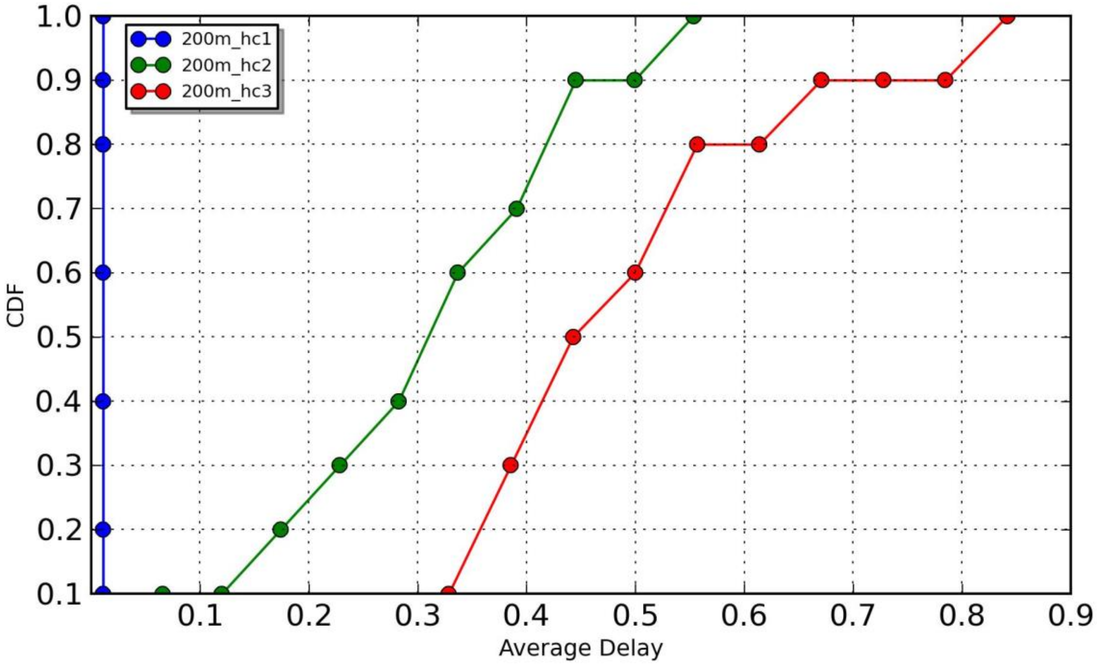


Figure 6.2: Cumulative Distribution Function for Average Hop Count Delays, with Group Size of 5 and Range 200m

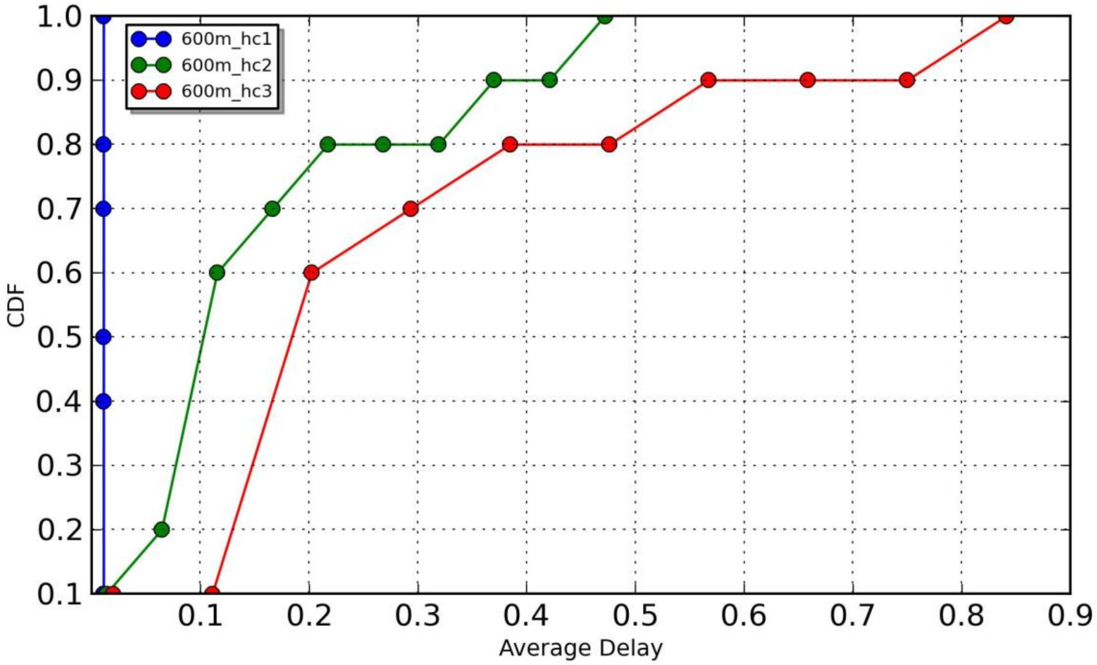


Figure 6.3: Cumulative Distribution Function for Average Hop Count Delays, with Group Size of 5 and Range 600m

Similar experiments are repeated for a group size of 3 nodes to analyze the impact of group size on delays. The results are show in Figures 6.4, 6.5 and 6.6. We witness lower delays with lower group sizes. This is because, as the group sizes reduce there are fewer chances for a node to move further away from the group members. Though the maximum distance of separation between some of the group members were greater than 1600m, there were few such cases in a group size of 3.

Similar to the delay statistics of group size 5, as the transmission range is increased from 200m to 600m the hop count delay drops down significantly. The CDF plots are shown in Figures 6.5 and 6.6 to prove this observation.

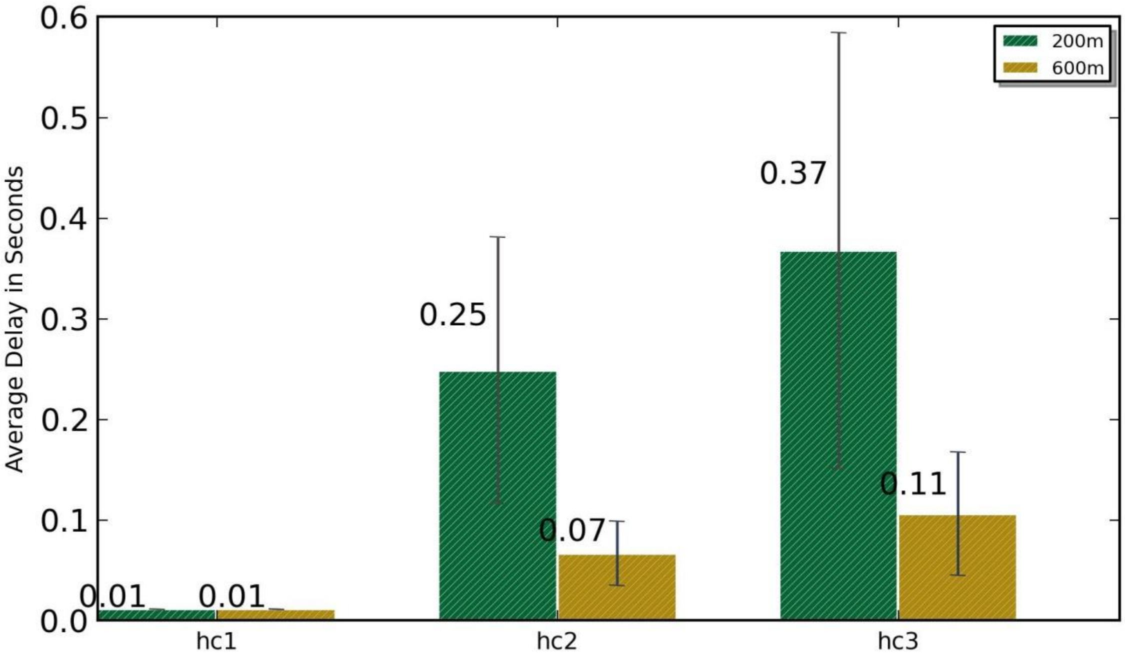


Figure 6.4: Average Delay Group Size 3

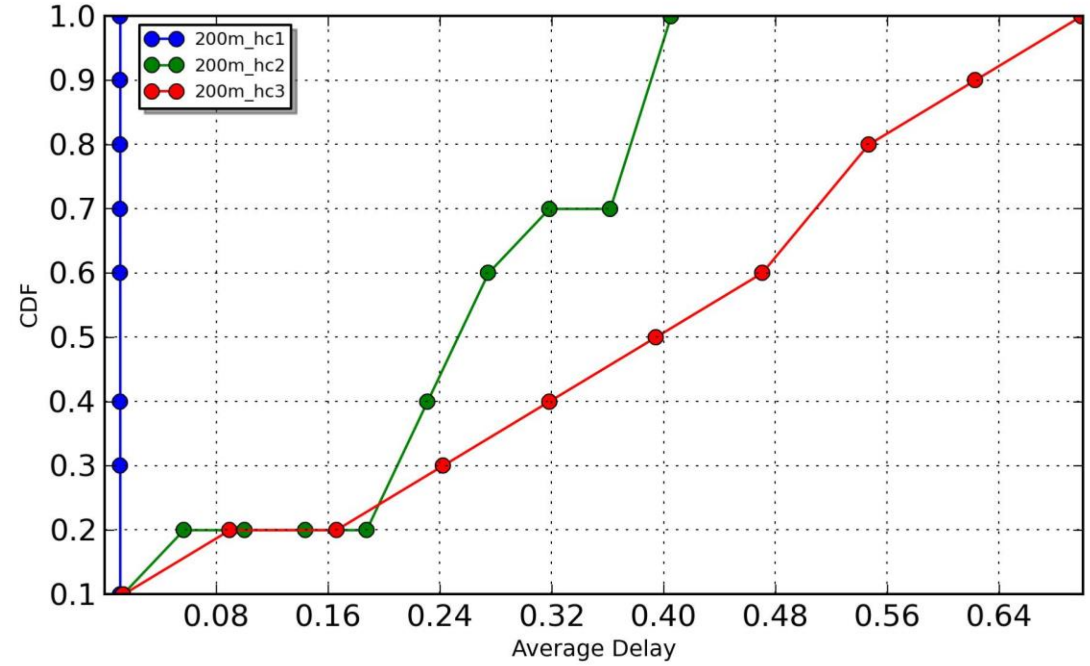


Figure 6.5: Cumulative Distribution Function for Average Hop Count Delays, with Group Size of 3 and Range 200m

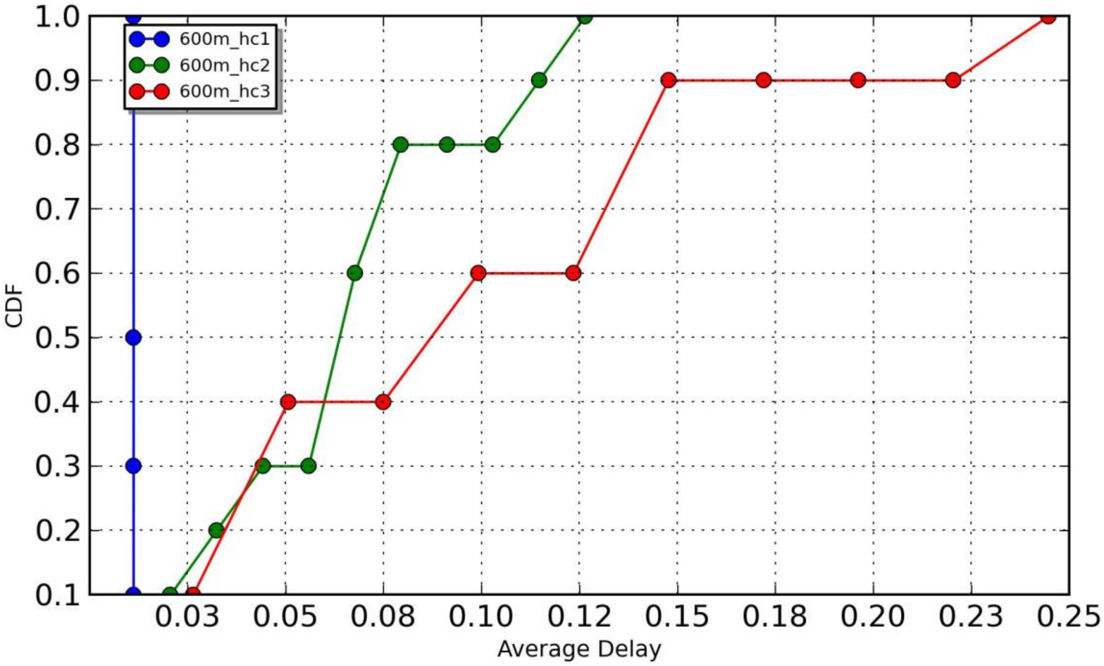


Figure 6.6: Cumulative Distribution Function for Average Hop Count Delays, with Group Size of 3 and Range 600m

6.2 DELAY STATISTICS FOR SBA BROADCAST

In this section we analyze SBA's performance on end to end delay (the delay between the broadcasting node and its group members), and compare it with the delay performance of Flood broadcast algorithm.

There is a strong correlation between the broadcast packet origination rate and the end to end delay [12]. The packet origination rate in our chat application varies with respect to the talk spurt duration. The end to end delay increases with the increase in packet origination rate; however, the content delivery ratio drops down. As mentioned in section 2.4.2, the percentage of area covered decreases with the increase in redundancy. Since SBA greatly reduces the redundancy, we can see a 35% drop in the delay value for hop count 2 and a 20% drop in the delay value for hop count 3 from the Figure 6.7.

When the range is increased to 600m, we do not find much difference in the end to end delay value between the Flood and SBA broadcast. The reason for this is that with increased range both protocols have more area to cover in each successive rebroadcast. Furthermore, more packets may be received within 2 hop counts instead of getting forwarded until 3 hop counts. From our experiments, it is clear that increase in transmission range plays an important role in reducing the end to end delay, and increasing the content delivery percentage.

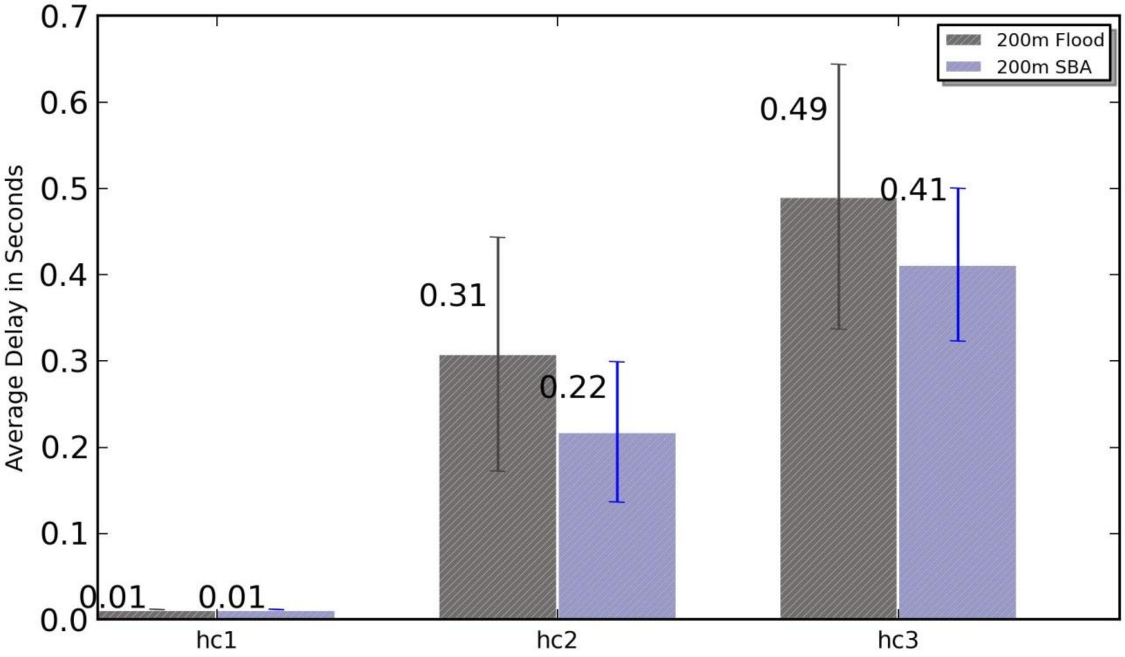


Figure 6.7 Average Delay, Flood Versus SBA, range 200m, Group Size 5

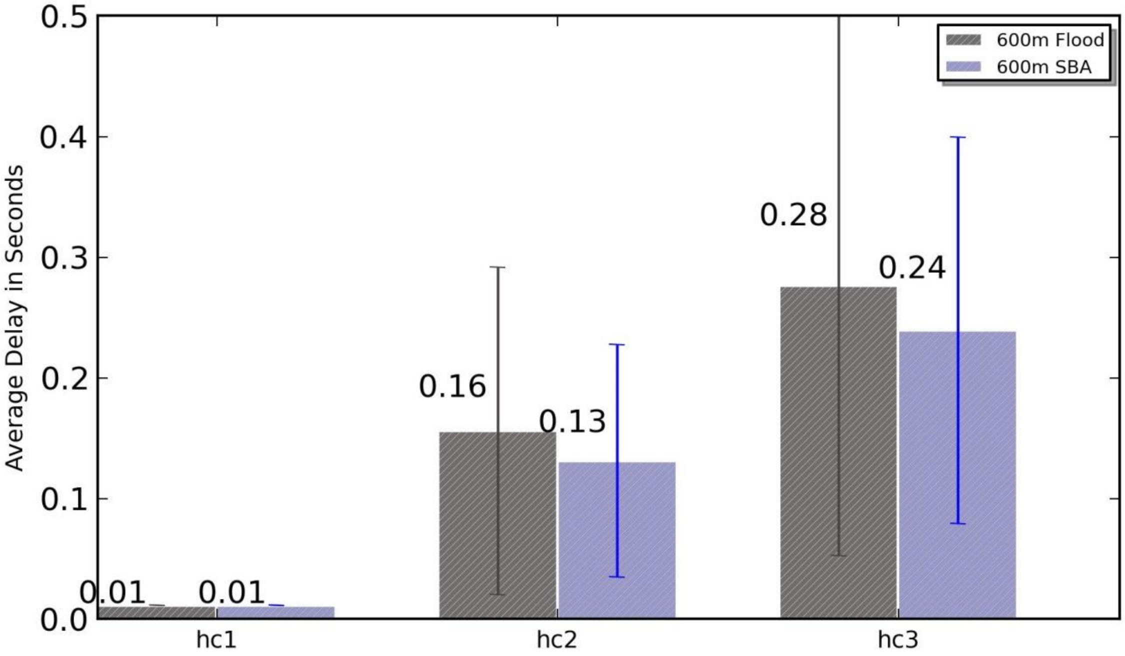


Figure 6.8 Average Delay, Flood Versus SBA, range 600m, Group Size 5

The CDFs for delays are shown in figures 6.9 and 6.10. From these figures we can clearly see difference in the delays when compared to the broadcast by Flooding technique. For 200m range most of the delay points 0.6 cdf with a maximum delay value of 0.26 whereas for Flood broadcast the values were 0.32 at 0.6 cdf. This trend is also quite apparent in hop count 3 scenarios where most of the delay points are concentrated within 0.8 cdf with the maximum delay value of 0.5 seconds. So we can see that the probabilities of obtaining an end to end delay between 0.2 and 0.25 are quite high in case of hop count 2 and 0.4 to 0.45 for hop count 3. Figure 6.8 shows the similar plot for a transmission range of 600m as explained before, the increase in the transmission range narrows down the difference in delay between the Flood and SBA broadcast. This observation is further confirmed by the figure 6.10.

The impact of varying group sizes on hop count delays are shown in figures 6.11 and 6.12. It is important to note that as the number of group members reduces, the difference in delays between Flood and SBA gets amplified. From figure 6.11 we can see that for hop count 2 we have about a 40% drop in the delay value from Flood to SBA broadcast and about 35% drop in hop count 3 scenario. The entire group size 3 scenarios have very low delay values due to reduced number of redundant packet received. Similar to above observations figure 6.12 shows the delay results for 600m range, where the delays have almost become negligible, figures 6.13 and 6.14 shows the cdf plots for a group size of 3.

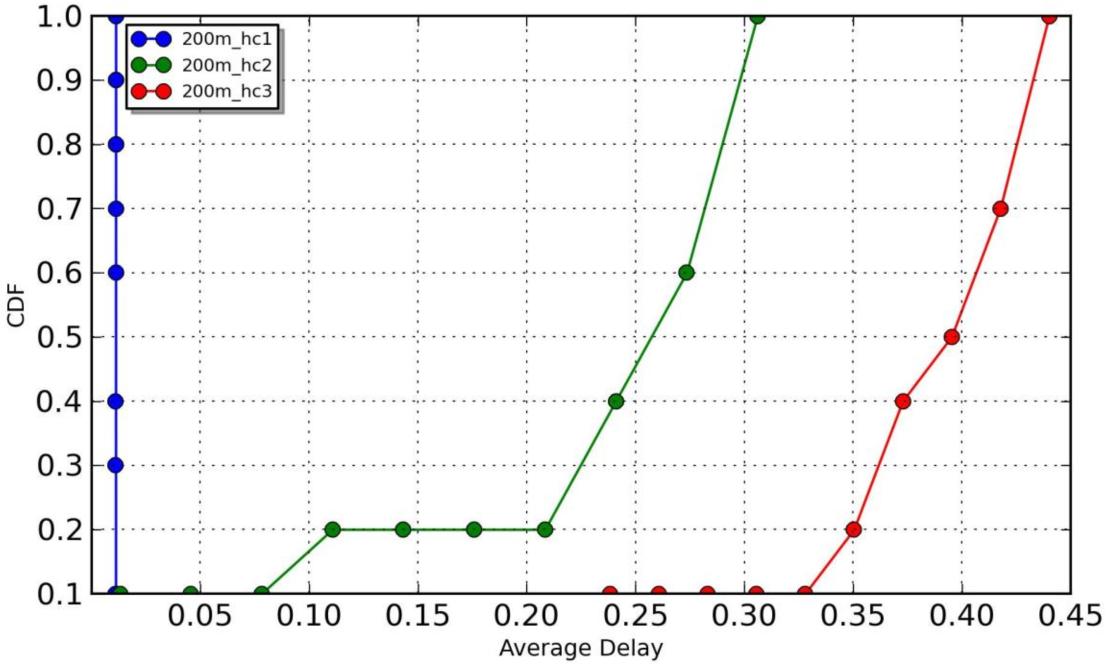


Figure 6.9: Cumulative Distribution Function for Average Hop Count Delays, with Group Size of 5 and Range 200

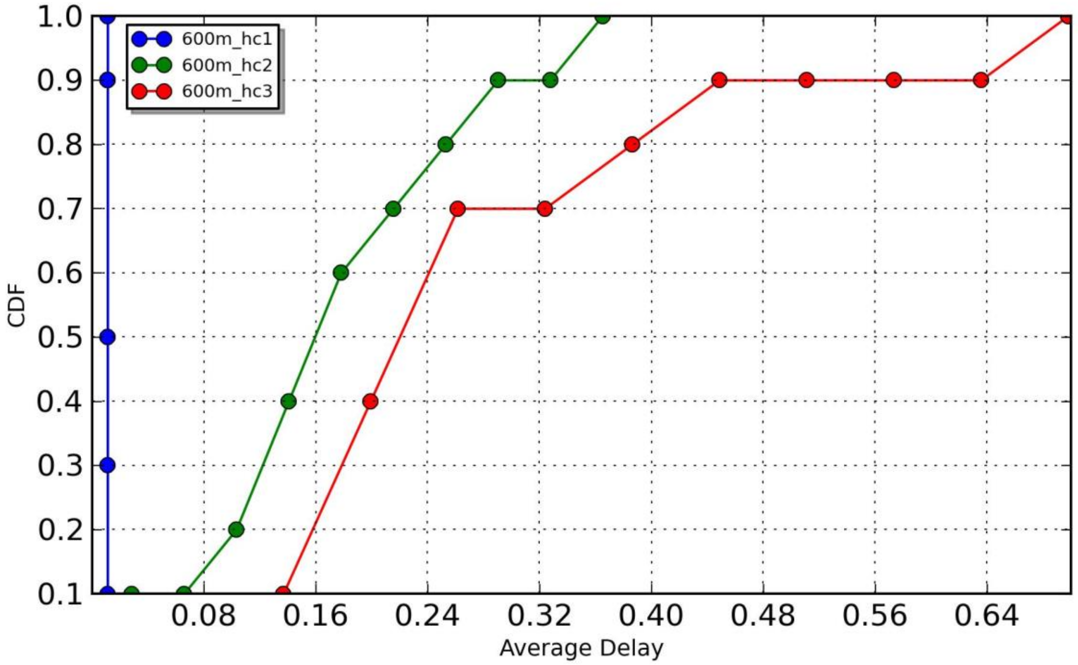


Figure 6.10: Cumulative Distribution Function for Average Hop Count Delays, with Group Size of 5 and Range 600m

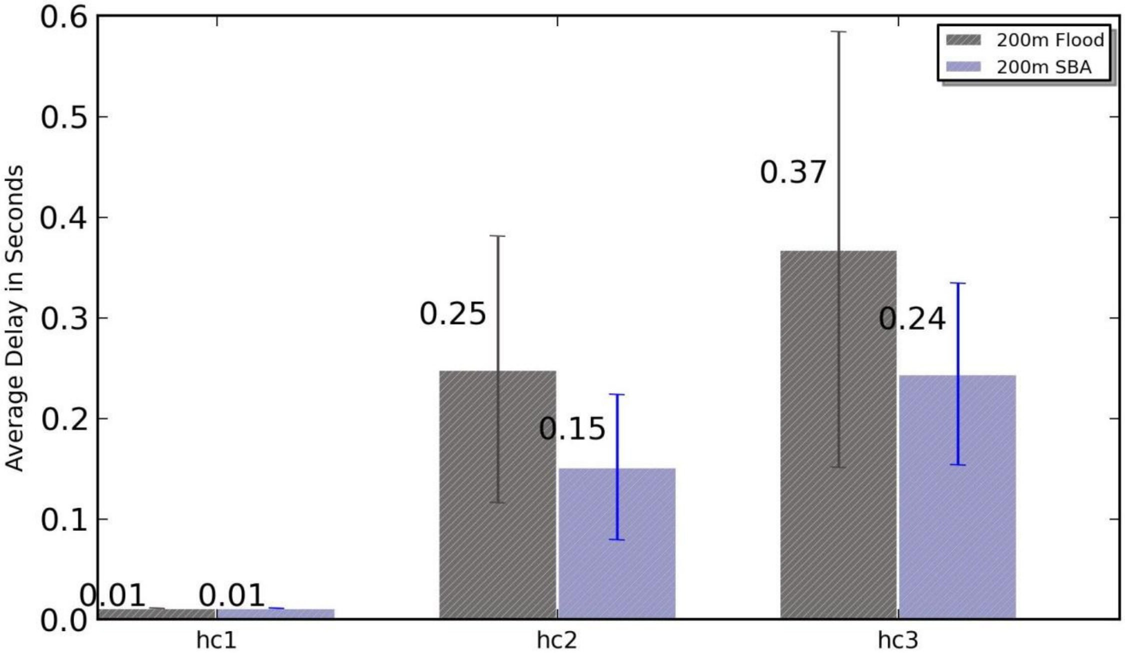


Figure 6.11 Average Delay, Flood Versus SBA, range 200m, Group Size 3

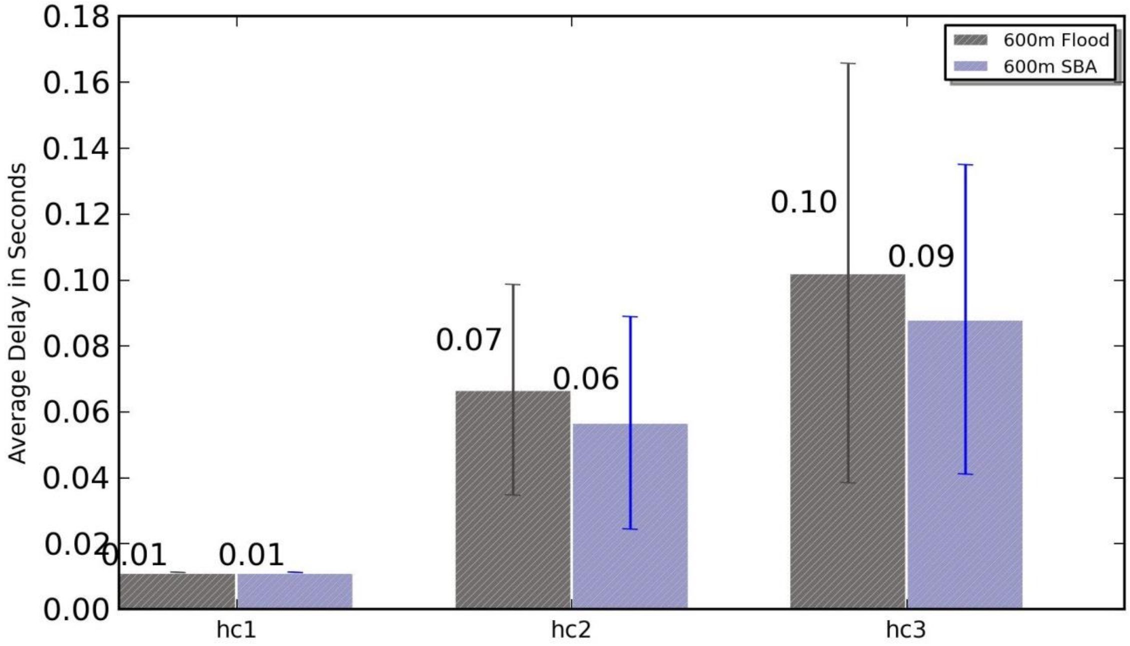


Figure 6.12 Average Delay, Flood Versus SBA, range 600m, Group Size 3

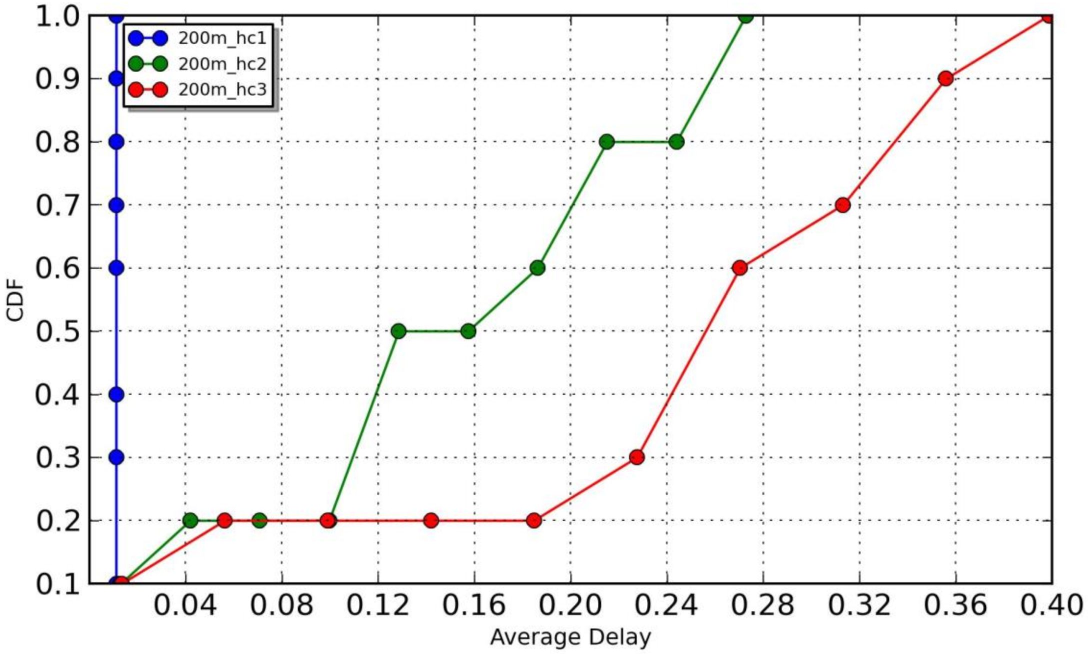


Figure 6.13: Cumulative Distribution Function for Average Hop Count Delays, with Group Size of 3 and Range 200m

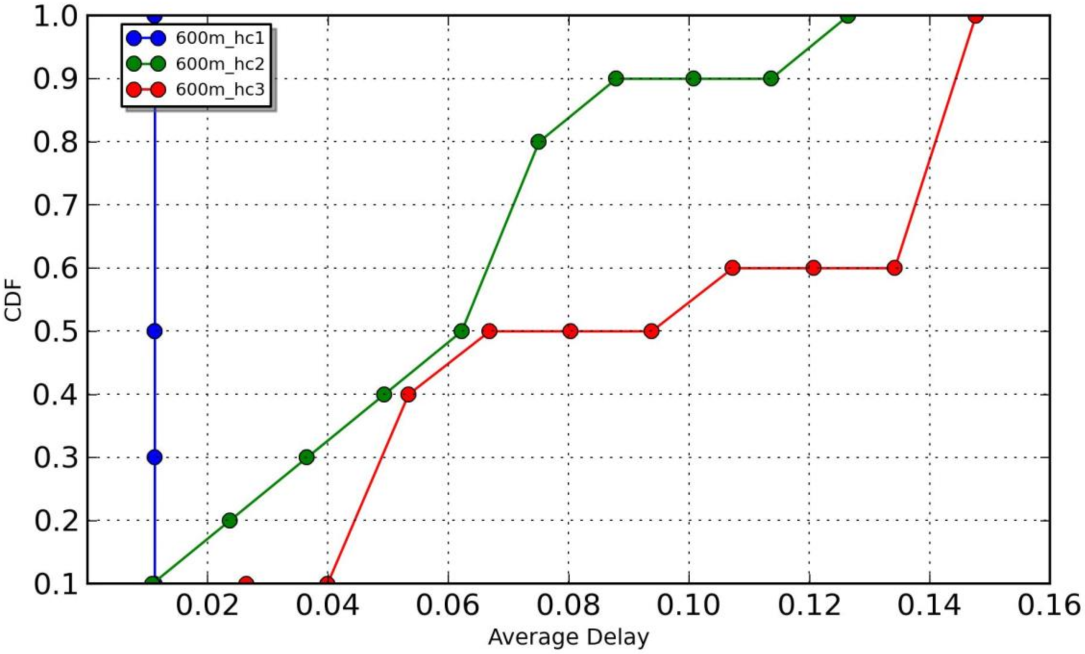


Figure 6.14: Cumulative Distribution Function for Average Hop Count Delays, with Group Size of 3 and Range 600m

6.3 EFFECT OF VEHICLE DENSITY ON AVERAGE DELAY

The increase in the number of node density results in improved forwarding and less packet loss. Even with a range of 200m the vehicles are able to find more forwarders; therefore, the end to end delay between the source node and the destination nodes (group members) is lower than the delay in low density scenarios. This effect is quite significant when the wireless radio range is set to 200m (Figure 6.15). When the range is increased to 600m, there is no significant difference in the delay values since the increased range gives better chances for a node to find its forwarder or destination nodes, irrespective of the increase in node density. Figure 6.16 clearly shows this effect, while Figure 6.17 and 6.18 shows similar results for a group size of 3.

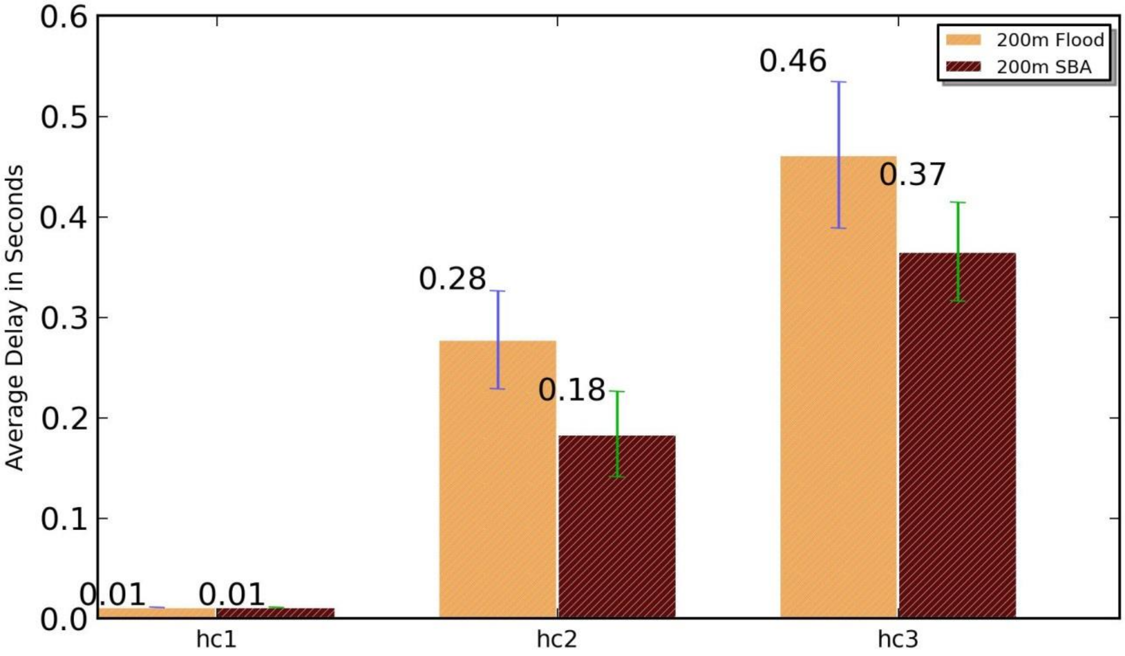


Figure 6.15: Average Delay, Flood Versus SBA, range 200m, Group Size 5, 1200vph

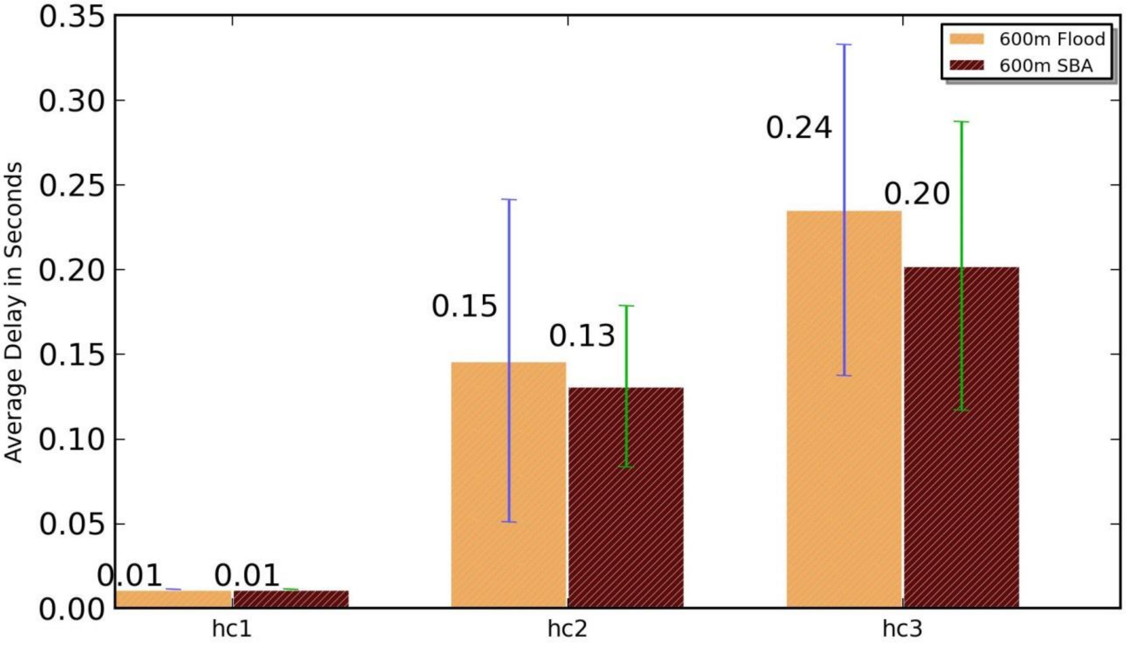


Figure 6.16: Average Delay, Flood Versus SBA, range 600m, Group Size 5, 1200vph

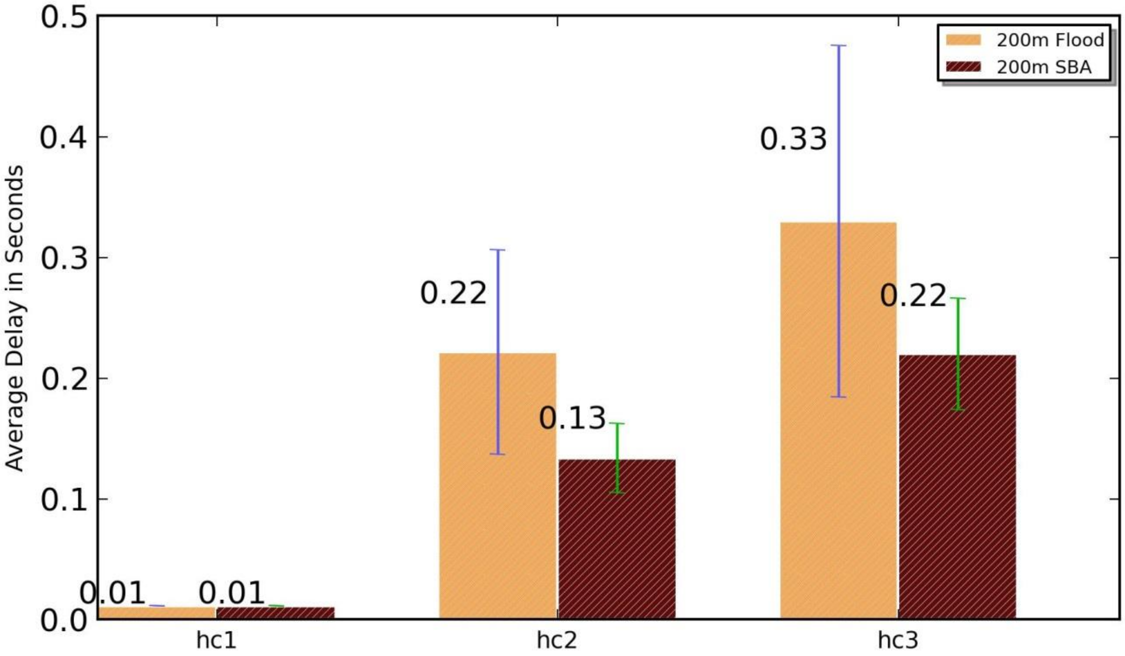


Figure 6.17: Average Delay, Flood Versus SBA, range 200m, Group Size 3, 1200vph

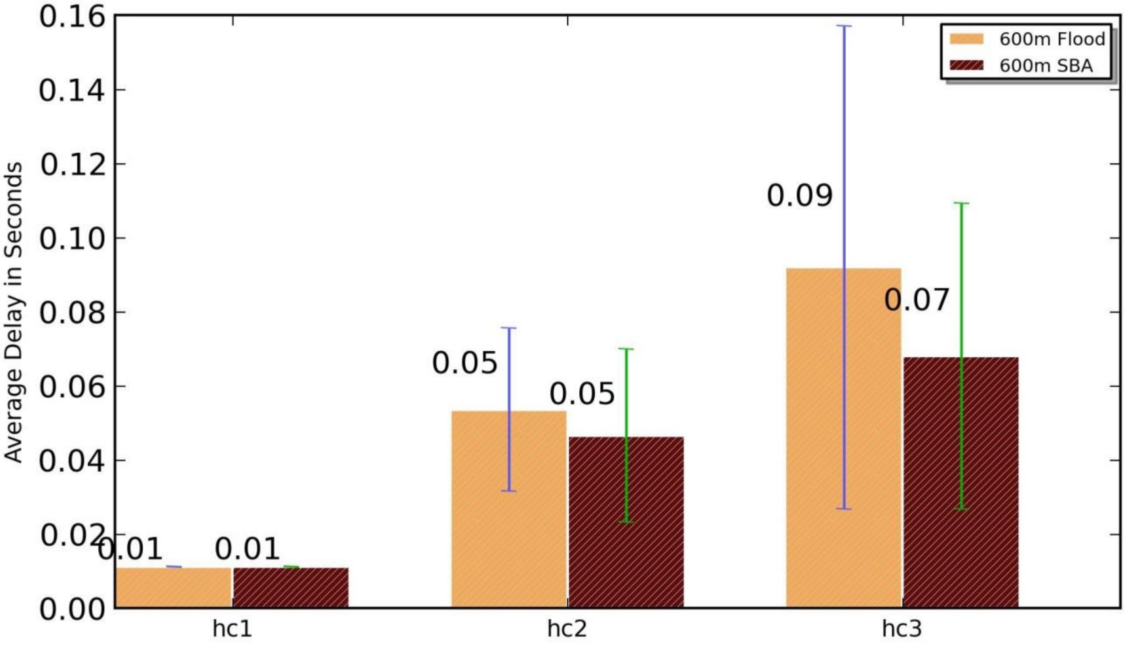


Figure 6.18: Average Delay, Flood Versus SBA, range 600m, Group Size 3, 1200vph

7 GROUP TALK BEHAVIOR

This section presents the results on the talk pattern between vehicles. The results of this section represent the voice chat application that we developed for inter-vehicle communication. Our main aim was to design a group chat algorithm that can mimic a casual talk between the group members. We wanted to make sure that none of the group members involved in voice chat scenario dominate the talk duration, like a class room lecture. From the results we can conclude that the designed algorithm is fair. This section is further divided into two subsections; section 7.1 illustrates the group talk behavior for Flood broadcast, and section 7.2 presents the results for group talk behavior based on SBA broadcast.

7.1 GROUP TALK BEHAVIOR – FLOOD BROADCAST

To analyze the performance of the group talk algorithm, we plot graphs that depict individual talking habits of group members (vehicles), which are involved in the chat scenario. Figure 7.1 shows the vehicle talk state graph for a group size of 5. On the x axis we have the simulation time which goes up to 1800sec, and on the y axis we have the vehicle talk states with values 0, 1 and 2. The y axis is extrapolated up to 5 units in order to make the figure clear. The state 0 implies that the vehicle (node) is at silent state (the node is not talking); 1 implies that the vehicle is at tentative talk duration and 2 implies the stable talk duration. Overall we can consider that a node begins to talk if it reaches the tentative talk duration.

From the Figure 7.1 and 7.2, we can see that all the group members reach the stable talk duration at regular intervals. As described in Section 2.3, the silence and talk spurts (talk duration) are modeled as Weibull distribution; the average length of the talk spurts were found to be less than 9sec from our experiments. It is important to note that none of the nodes dominate the group talk environment. We show this result in Figure 7.2 in which the clock state 2 denotes the talk duration. The figure represents frequent peaks of talk spurts that are distributed throughout the length of the simulation. We can also observe some lengthier silence duration, but none of these durations are too long. This means that a node actively starts broadcasting the packets after brief silence duration. Similarly, there are no continuous talk spurts with very long duration.

During a casual group chat it is quite natural for a person to interfere while the other person talks. Similar behavior is observed in our voice chat model; we call this as the simultaneous talk scenario, which is presented in Figure: 7.2. The maximum duration for which two nodes simultaneously talk is about 2 seconds. We observed the same trend over all the 10 groups that we chose for simulation.

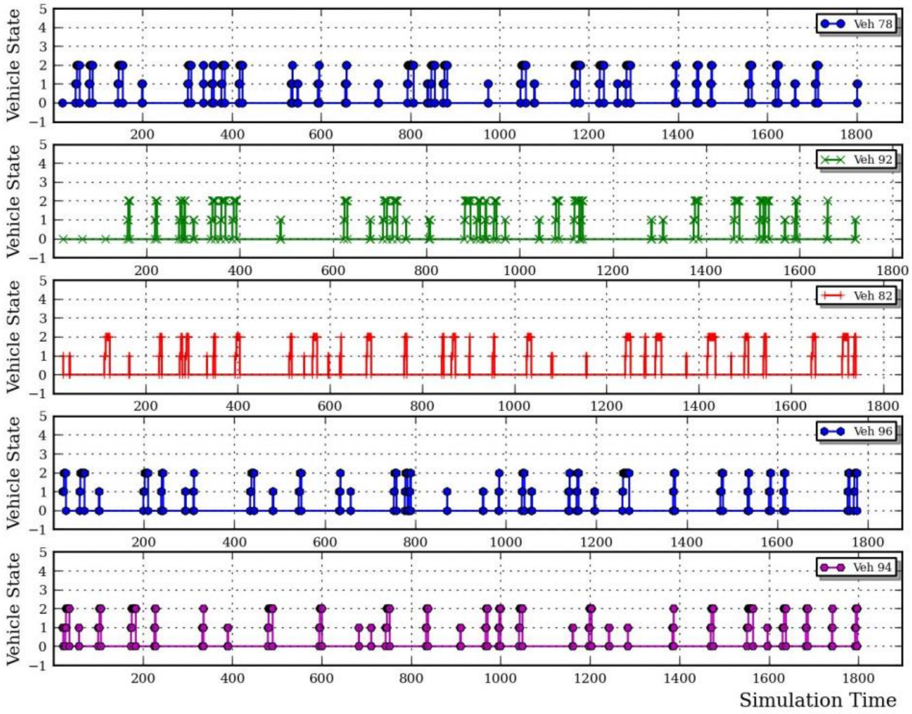


Figure 7.1: Vehicle Talk State Group Size 5

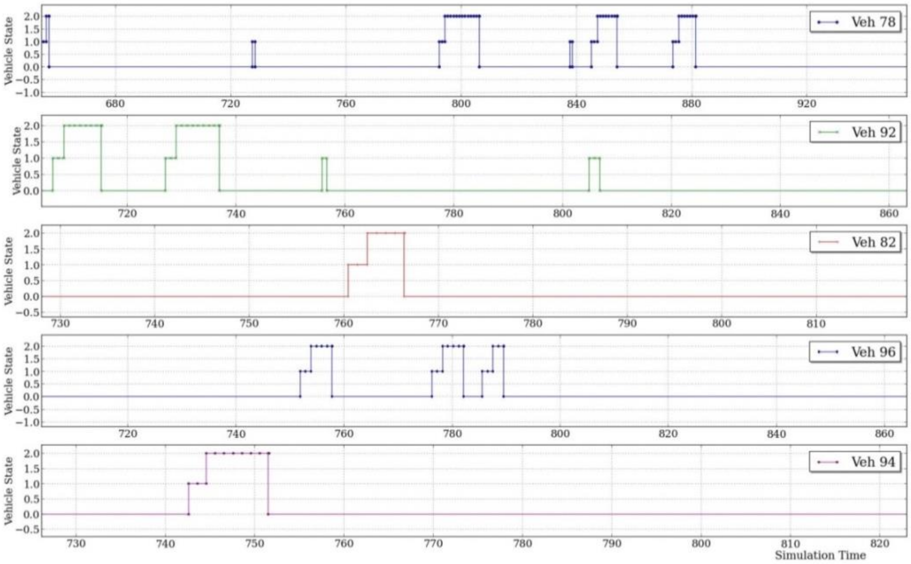


Figure 7.2: Simultaneous Talk Scenario Group Size 5

We repeated the same set of experiments by varying the group size from 5 to 3 vehicles. The outcomes of these experiments are shown in figures 7.3 and 7.4. We see that the talkspurts follow similar pattern as that of vehicle talk state with a group size of 5. The major difference between Figure 7.1 and Figure 7.3 is the frequency of talk spurts. Since the number of vehicles in a group is reduced from 5 to 3, Figure 7.3 shows less number of talk spurt peaks when compared to Figure 7.1.

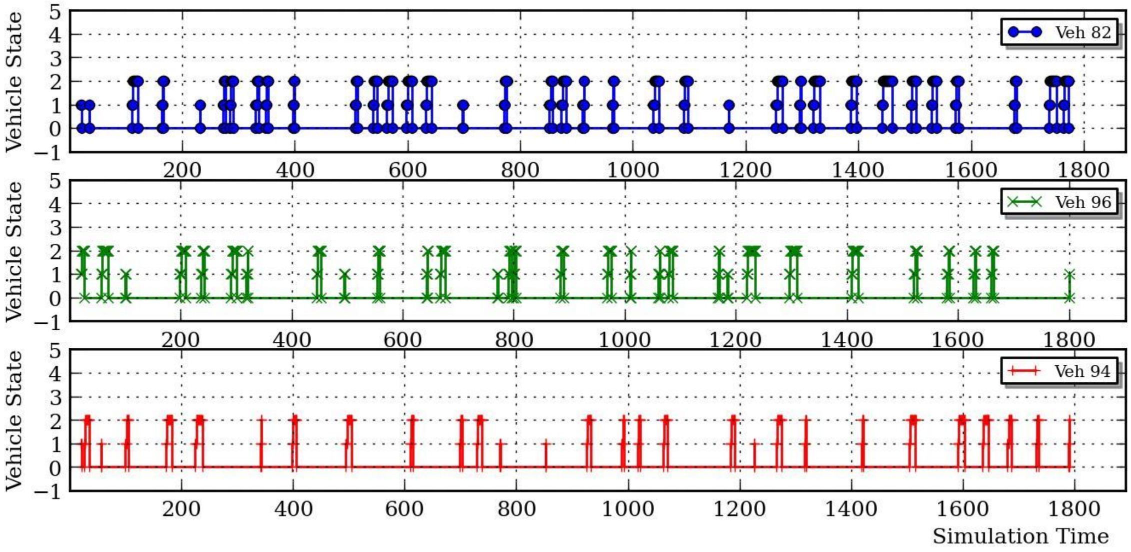


Figure 7.3: Vehicle Talk State Group Size 3

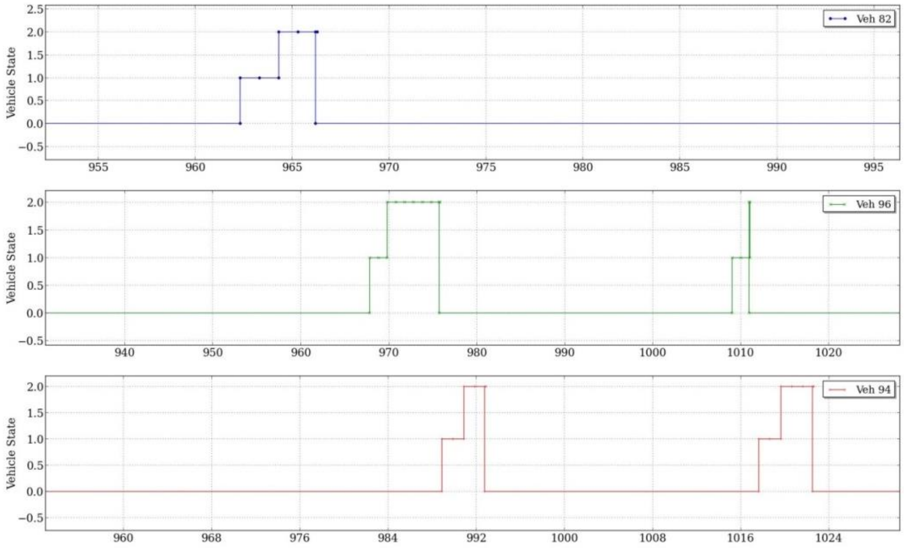


Figure 7.4: Maximum Talk Duration Group Size 3

7.2 GROUP TALK BEHAVIOR – SBA BROADCAST

In this section, we show the results of running the voice chat algorithm using SBA broadcast scheme. Figure 7.5 shows that there is no difference in the group talk behavior for SBA when compared to the Flood broadcast. This means that by changing the underlying broadcast protocol the talking behavior does not change. Furthermore, we also notice that the fairness of the algorithm remains constant irrespective of the change in the broadcast scheme. We observe short *talkspurts* which are evenly distributed over the entire simulation duration; also it is quite visible that none of the nodes dominate the entire talk duration. This observation is quite similar even with varying group sizes. Figure 7.6 shows the result of the group chat behavior with a group size of 3, which substantiates the fact that the change in the Broadcasting Scheme has little effect over the group chat habits.

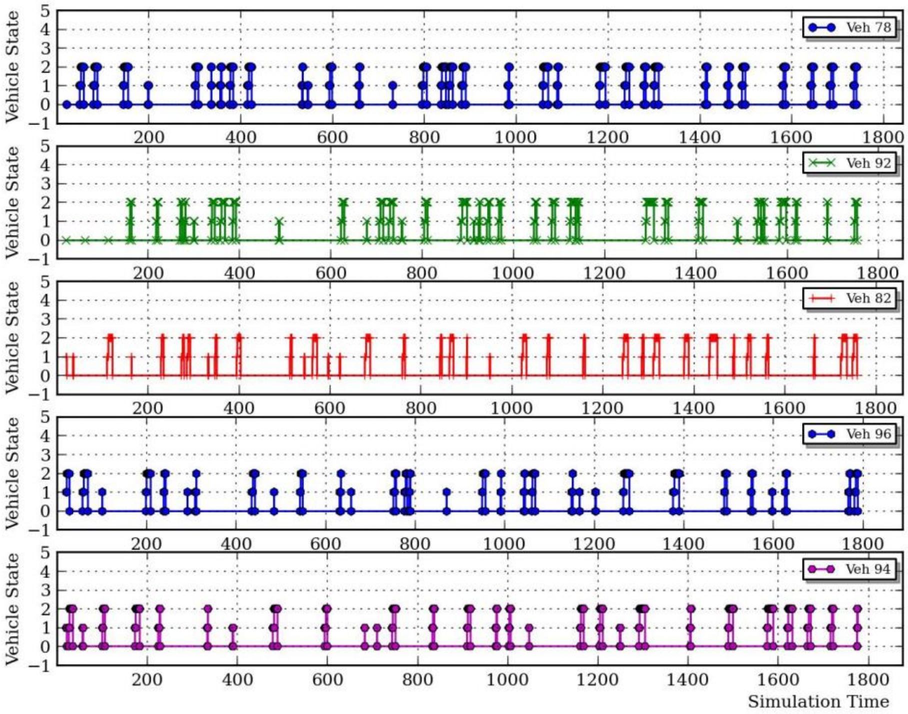


Figure 7.5: Vehicle Talk State Group Size 5 – SBA

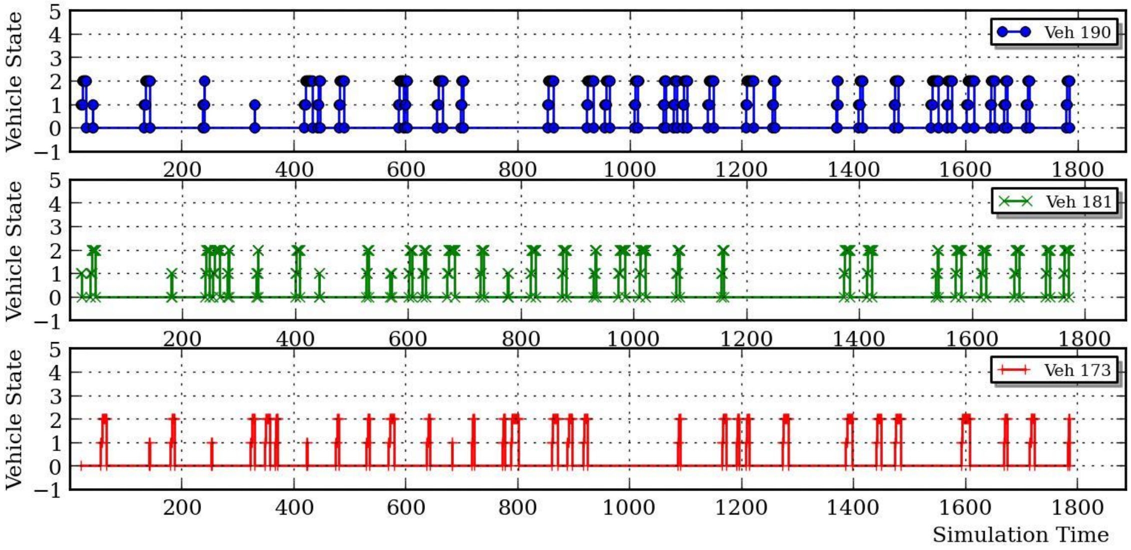


Figure 7.6: Vehicle Talk State Group Size 3 – SBA

7.3 EFFECT OF NODE DENSITY ON GROUP TALK BEHAVIOR

The effect of increase in the vehicle density is shown in figures 7.7 and 7.8. The increase in vehicle density results in lower packet loss, and better communication between the group members. This in turn results in frequent talk spurts, which is shown as dense peaks in Figure 7.7 and 7.8.

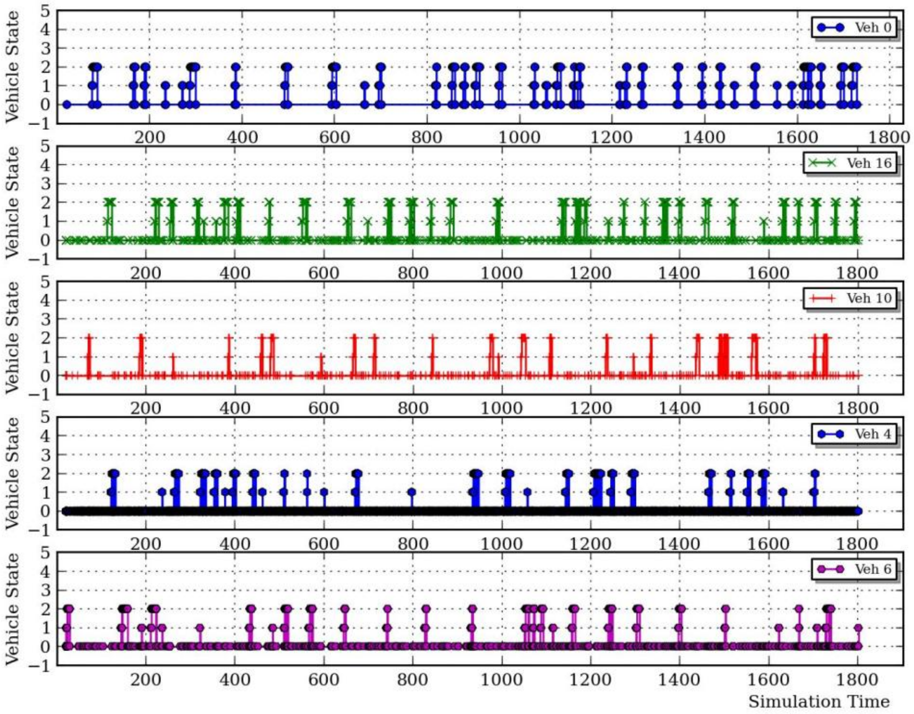


Figure 7.7: Vehicle Talk State Group Size 5 – FLOOD

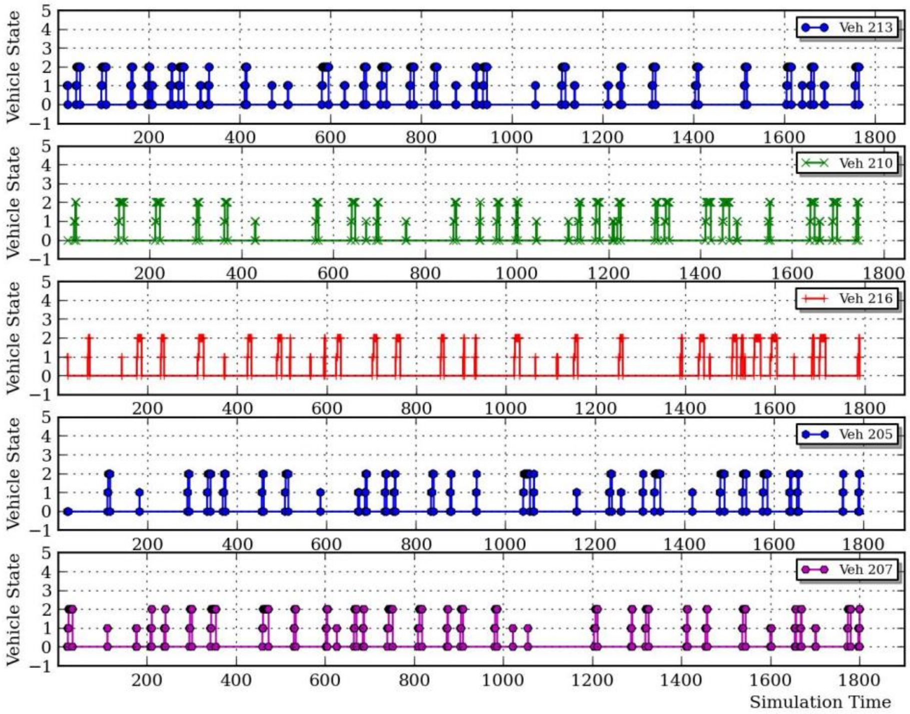


Figure 7.8: Vehicle Talk State Group Size 5 – SBA

8 NODE FAIRNESS VALUE

This section describes the behavior of voice chat application by presenting the fairness values of different nodes with varying group sizes. As stated in Section 2.3 fairness value is defined as the claim to speak. When a node has higher fairness value then it has more claim to speak. During this time if any other node interrupts the currently publishing node then the node with higher fairness claims the right to talk and sends the other node to silence. This section is divided into two sub sections, section 8.1 presents the group talk behavior for the Flood algorithm and section 8.2 presents the group talk behavior for SBA algorithm.

From Figure 8.1 we find that there is a steady decline in a node's fairness value

for a group size of 5 and a maximum hop count of 1, this is because of the fact that the fairness value is reduced by 2 when a node finishes talking. It is interesting to observe that no node dominates the group chat scenario since there is no sudden increase in the fairness value of nodes. There is a small increase in fairness value for every node in a group throughout the simulation duration and every node eventually approaches the lowest fairness value of 0 along with the simulation duration. It is also important to note that once the nodes reach their minimum fairness value they do not climb back to a huge fairness value in case of hop count one. But for hop counts two and three we observed a different trend. From figures 8.2 and 8.3 we observe that the fairness values do not reach the minimum (a value of 0). We also find frequent oscillations between lower fairness to higher fairness value and vice versa. In the figure 8.2 though the vehicle 99 goes to the least fairness value of 0, we witness an even transition from the highest fairness value to the lower values for all the 5 node group members (vehicles). The frequent oscillations in fairness values can be attributed to the fact that with increase in the hop count the Content Received percentage is higher which means that the packet loss is low, which in turn implies the complete participation of all the vehicles in the chat scenario.

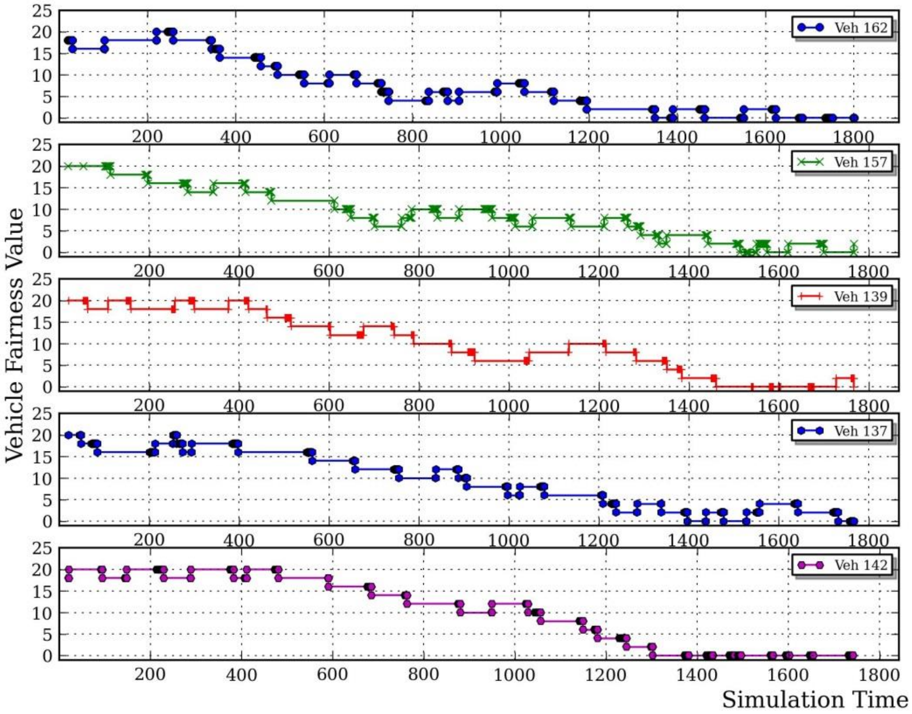


Figure 8.1: Node Fairness Value Group Size 5 Hop Count 1

This trend varies with respect to group sizes. Figures 8.4 and 8.5 show a smooth transition from higher fairness value to the lower fairness value for a group size of 3. It is also important to note that all the nodes reach their minimum fairness value of 0 irrespective of the increase in hop count. This means that as the group size is reduced the reachable set increases which implies that most of the group members can be reached within 2 hops, this is the reason we did not observe much of variation in the content received percentage between hop count 2 and 3 with a group size of 3. The change in the group size has also considerably altered the group chat behavior, as the group size decreases from 5 to 3 we observe a smooth talk scenario with a steady decrease in fairness value.

It is also important to observe the maximum increase in the fairness value. The maximum increase in the fairness value is observed to be 2 steps as shown in the Figure 8.6, after this increase the fairness value immediately comes down which means that the node has completed its talkspurt successfully. This 2 step increase in the fairness value was the maximum increase that happens for all the vehicles involved in the group chat. This increase is irrespective of the group size, hop count or range. However, the frequency of such increase is low in the case of hop count 1 while it is high for hop count 2 and 3.

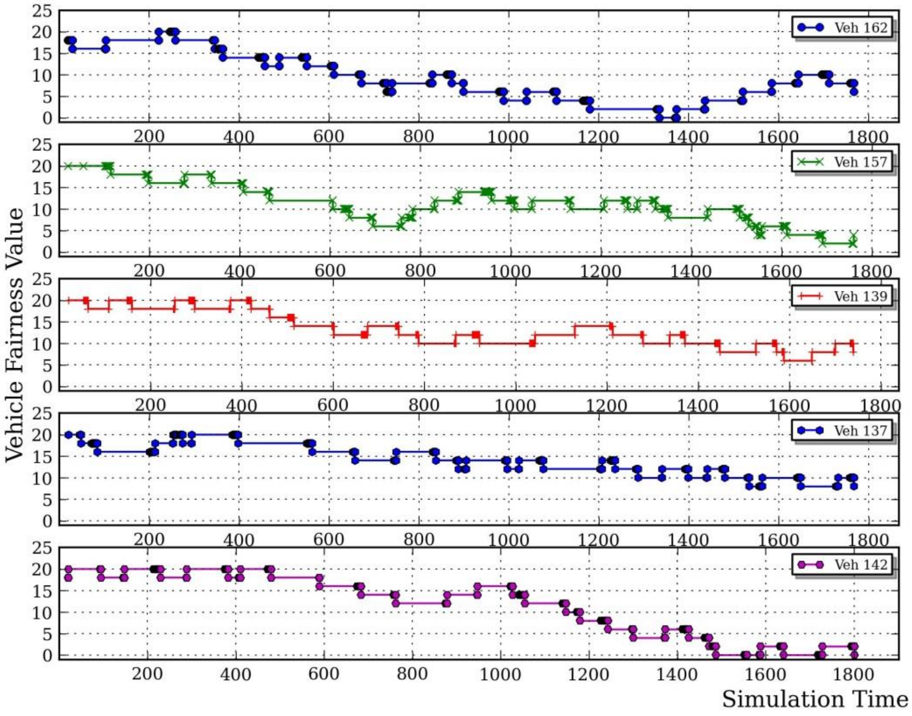


Figure 8.2: Node Fairness Value Group Size 5 Hop Count 2

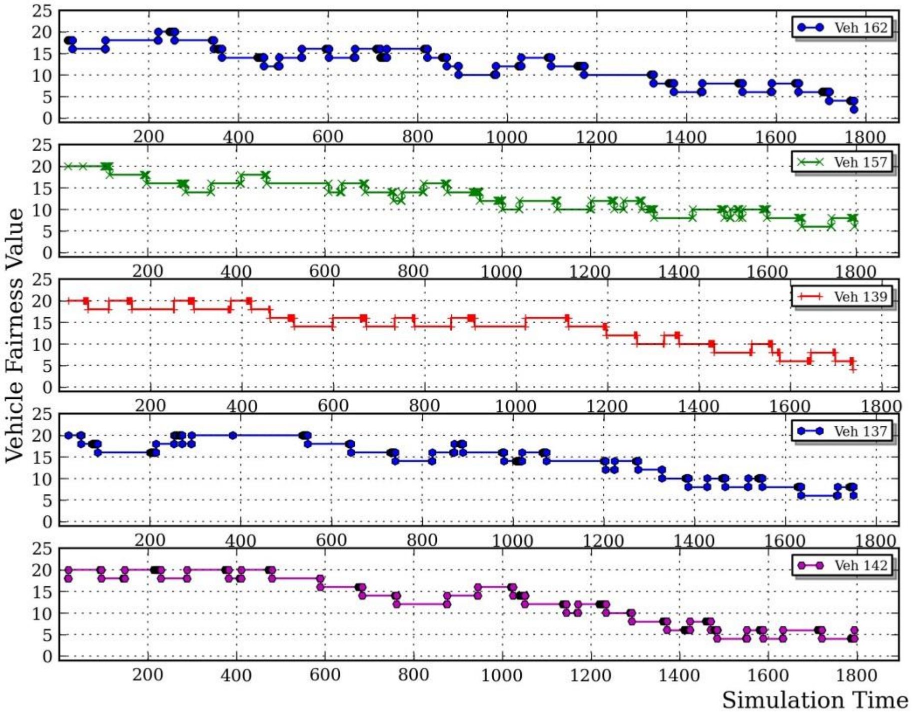


Figure 8.3: Node Fairness Value Group Size 5 Hop Count 3

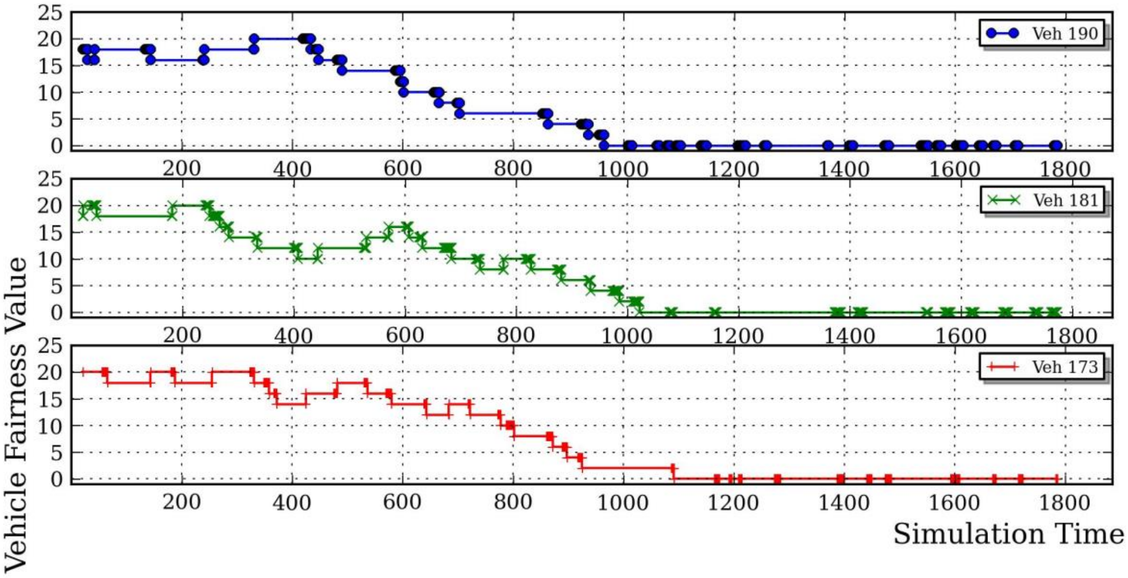


Figure 8.4: Node Fairness Value Group Size 3 Hop Count 1

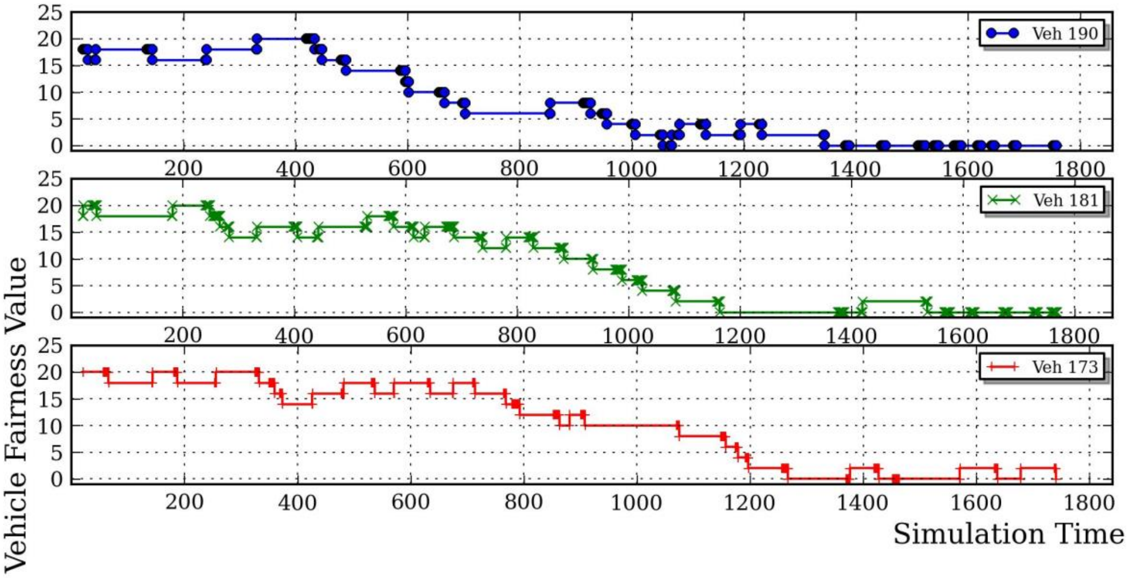


Figure 8.5: Node Fairness Value Group Size 3 Hop Count 3

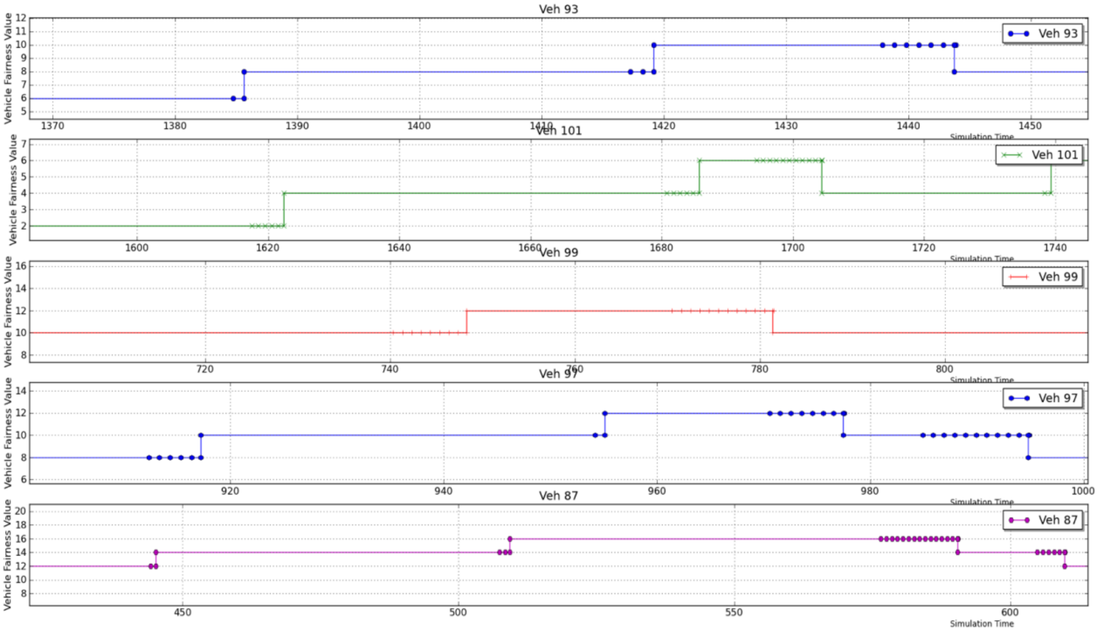


Figure 8.6: Maximum Increase in Fairness Value, Group Size 5 Hop Count 2

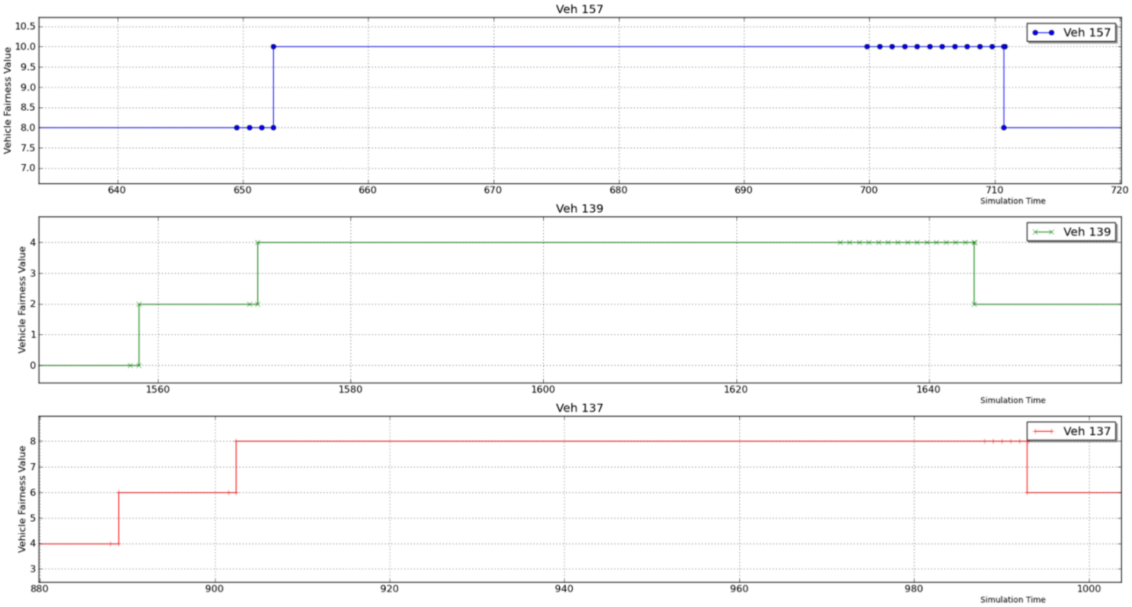


Figure 8.7: Maximum Increase in Fairness Value, Group Size 3 Hop Count 1

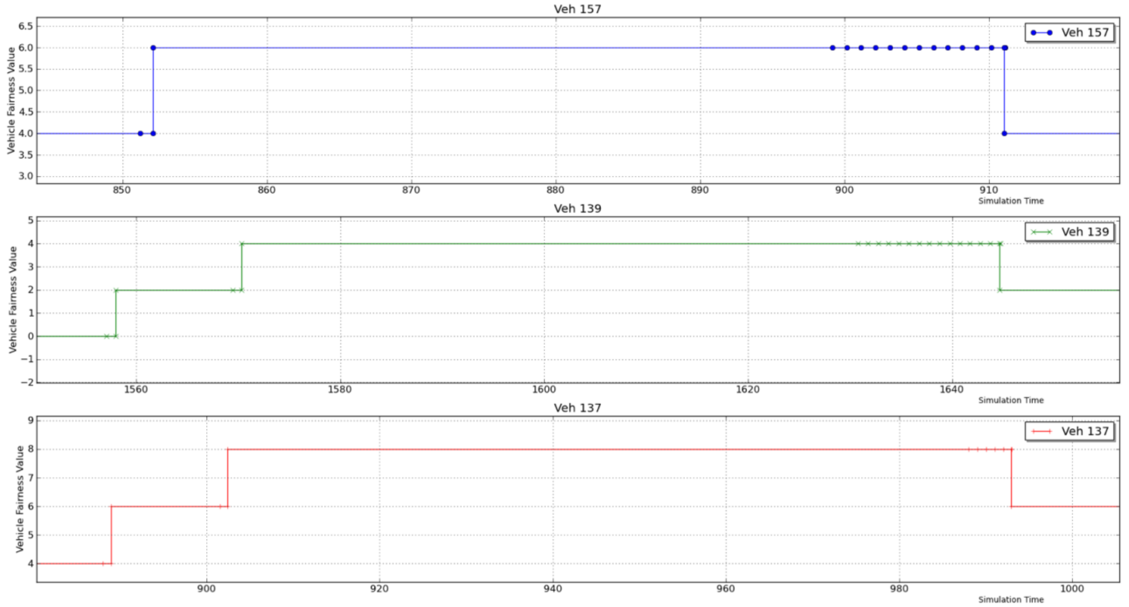


Figure 8.8: Maximum Increase in Fairness Value, Group Size 3 Hop Count 3

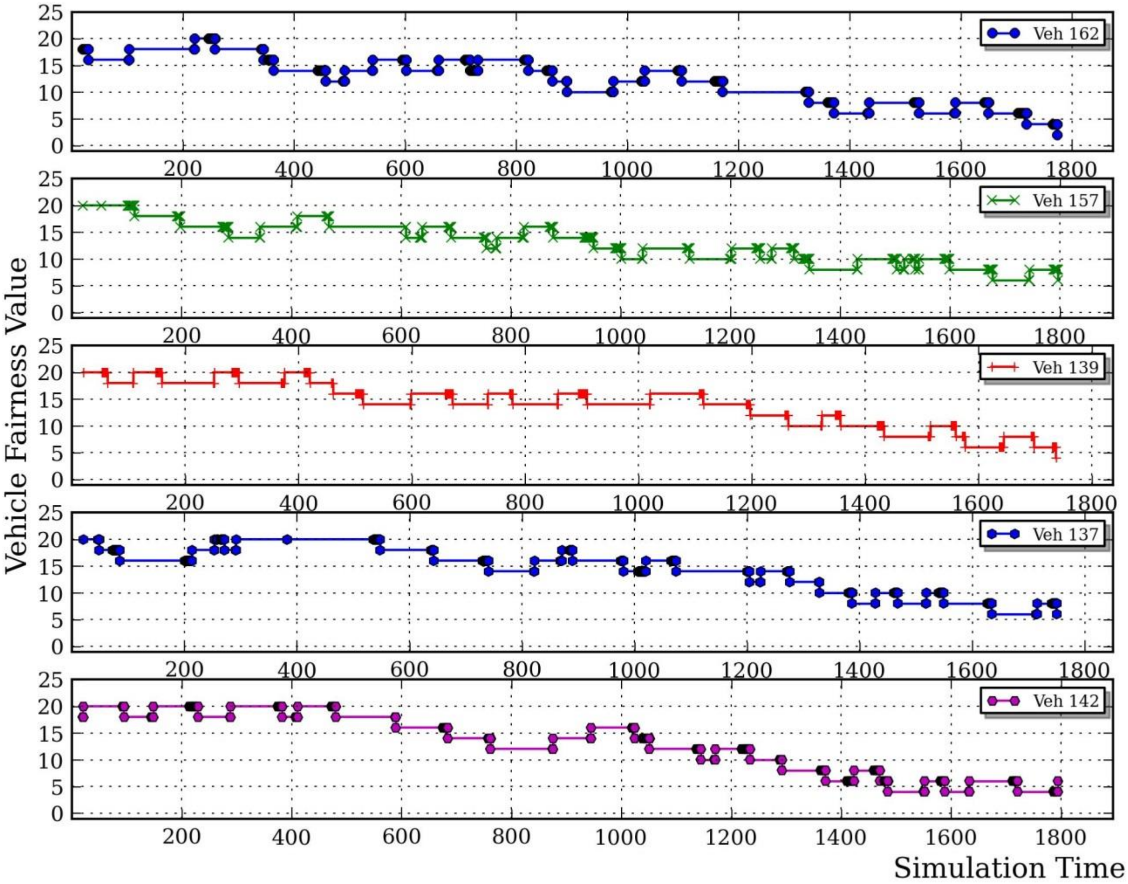


Figure 8.9: Maximum Increase in Fairness Value, Group Size 5 Hop Count 3 – SBA

9 INTERRUPTED NODES

This section delineates the effect of the proposed algorithm over the number of Interrupted nodes. Nodes get interrupted in two states, namely the Tentative Talk State and the Stable Talk State. A node that is currently broadcasting the packet (Talking) is interrupted by other node in the Stable Talk State if the interrupting node has higher fairness value than the currently broadcasting node. The node that is interrupting may get interrupted in the Tentative Talk State if the node that is currently broadcasting has more packets to send. In both the cases the interrupted nodes are sent back to Silent State which then follows a Weibull timer as Silence duration. The Section 9.1 illustrates the effect of Flooding broadcast over the interrupted nodes and Section 9.2 presents the same result with SBA as broadcast protocol.

9.1 INTERRUPTED NODES – FLOOD BROADCAST

Figure 9.1 illustrates the number of Interrupted nodes in a group size of 5. As the hop count increases the number of interrupted nodes increase since the number of published packets increase. The increase in the Interrupted node is quite discernible for hop counts one and two but the margin of increase slims down as we move from hop count 2 to 3. This observation is also noticed in the 600m transmission range scenario.

The effect of group size in interrupted node count is shown in Figure 9.2, where we can witness a significant decrease in the number of interrupted nodes. This observation is again due to the number of packets that are being published, group sizes of 5 have higher number of packets being broadcasted due to more number of group members hence we have more chances of a node getting interrupted or more chances of a two nodes talking simultaneously.

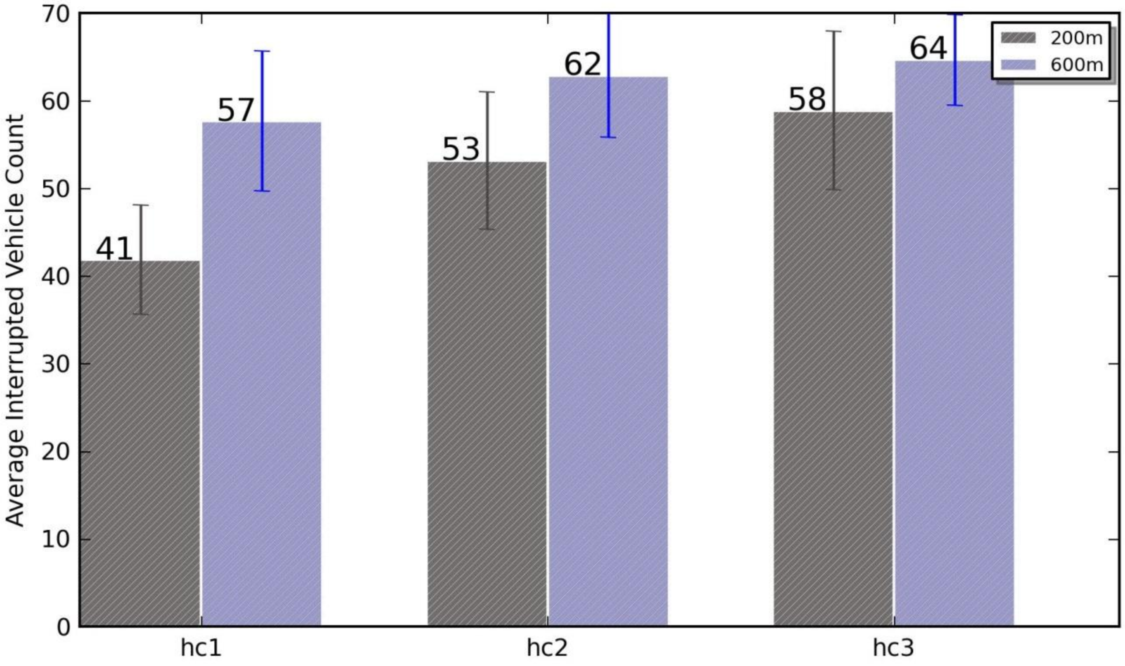


Figure 9.1: Interrupted Count Group Size 5

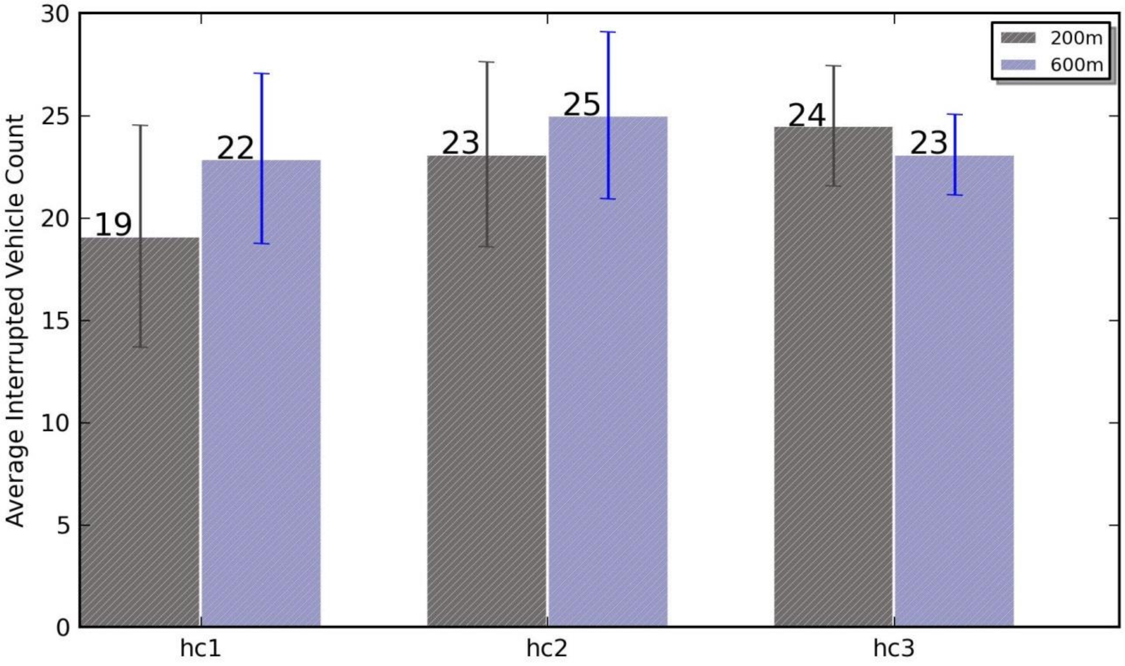


Figure 9.2: Interrupted Count Group Size 3

9.2 INTERRUPTED NODES – SBA BROADCAST

From the previous results we concluded that the Broadcast protocol does not affect the group chat behavior, to complement those results figure 9.3 shows that there is subtle variation in the number of Interrupted vehicles when compared to figure 9.1 which involved flooding broadcast. The variation in broadcasting technique has not resulted in a huge variation in the number of interrupted nodes since we did not find much of a variation is the fairness value pattern. The change in group size results in the same pattern as that of flood broadcast, this is shown in figure 9.4.

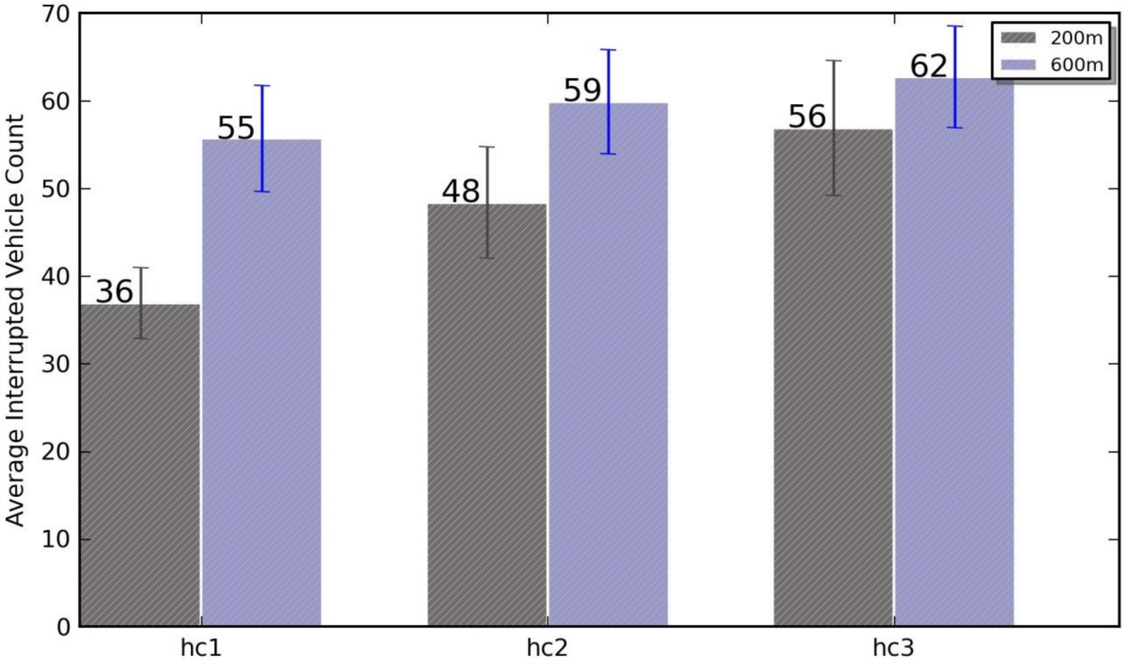


Figure 9.3: Interrupted Count Group Size 5 – SBA

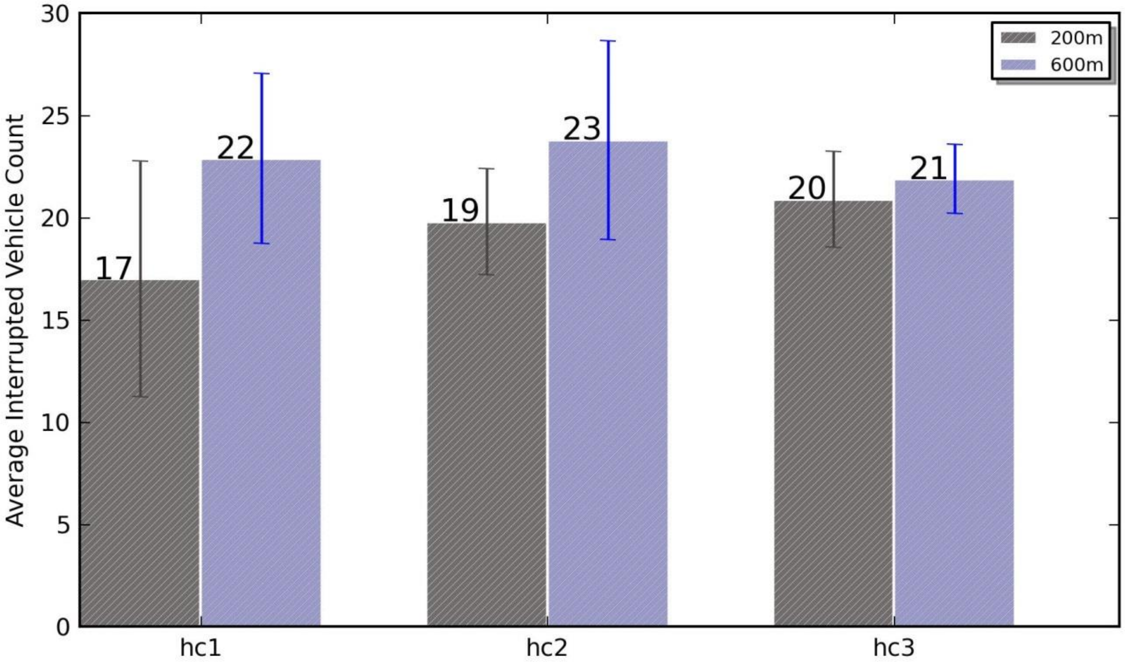


Figure 9.4: Interrupted Count Group Size 3 – SBA

10 JITTER DELAYS IN VOICE CHAT

Jitter is defined as a variation in the delay of received packets. Measuring Jitter in inter-vehicle voice communication is important to determine the quality of the voice chat. Jitter is caused by transit delay, contention in the network, and serialization effects. These issues take place on the pathway of the network that transfers the data from the source node to the destination nodes. In this research we measure two types of jitter: 1. for each pair of sender and receiver (S, R), we calculate jitters for the sequence of packets $\{P_1, P_2, \dots P_n\}$ from S to R; 2. for each packet P_j, $j = \{1,2 \dots n\}$ from a sender S to receiver R we calculate the difference in delays for the packet P_j to reach the individual desired receivers. The results of these two jitter measures are reported in figures 10.1-10.5. Figure 10.1 and 10.2 shows the cumulative distribution of jitter standard deviation. We can see that the SBA outperforms Flood in both hop counts 2 and 3; however, the difference between Flood and SBA is not very large. This trend is again captured by the CDFs of the coefficient of variation (CV) in plots 10.3 and 10.4. Furthermore, from Figure 10.5 we can observe that approximately 95% of the jitters lie within 0.7 seconds for both Flood and SBA when the hop count is set to 2. The increase in transmission range results in lower jitter delays, which is clearly captured in Figure 10.5 and 10.6.

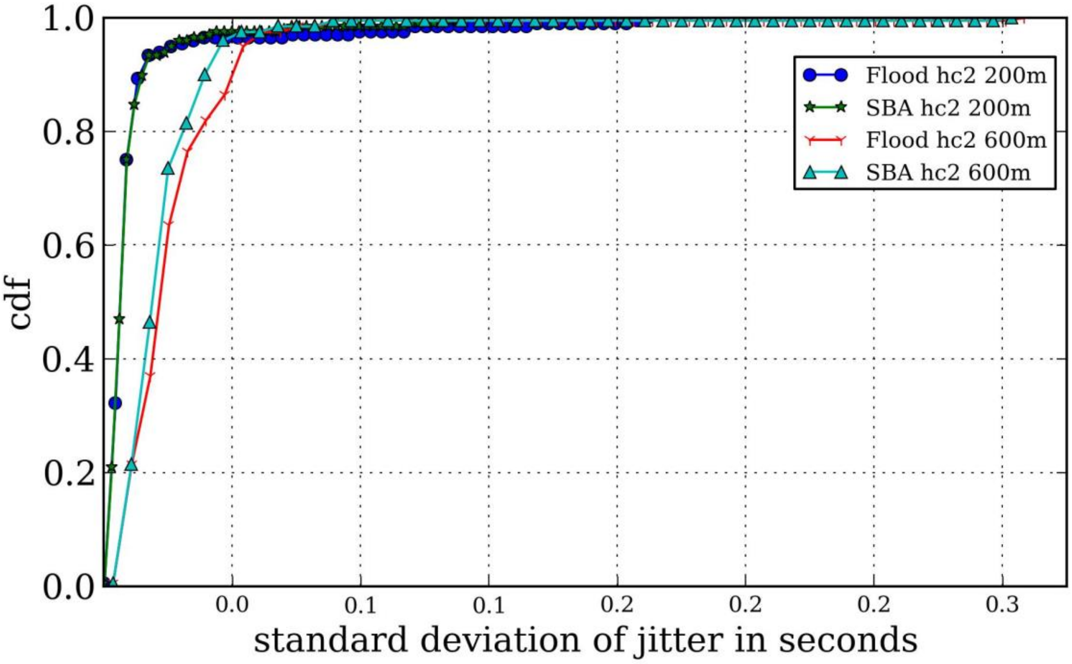


Figure 10.1: CDF of Jitter Standard Deviation for hop count 2, ranges 200m and 600m

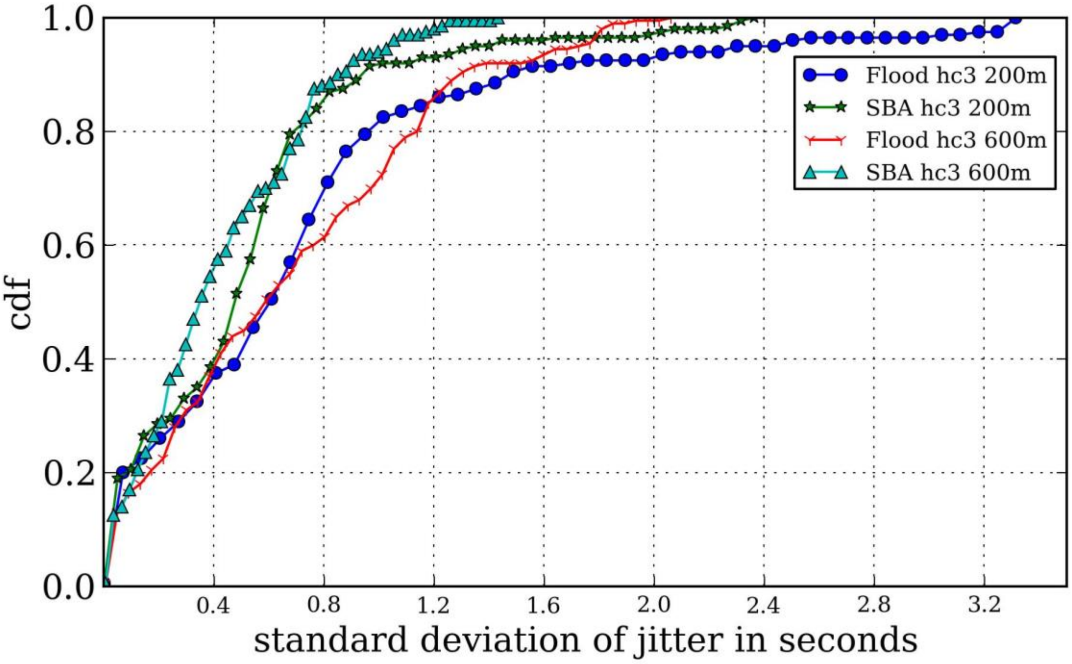


Figure 10.2: CDF of Jitter Standard Deviation for hop count 3, ranges 200m and 600m

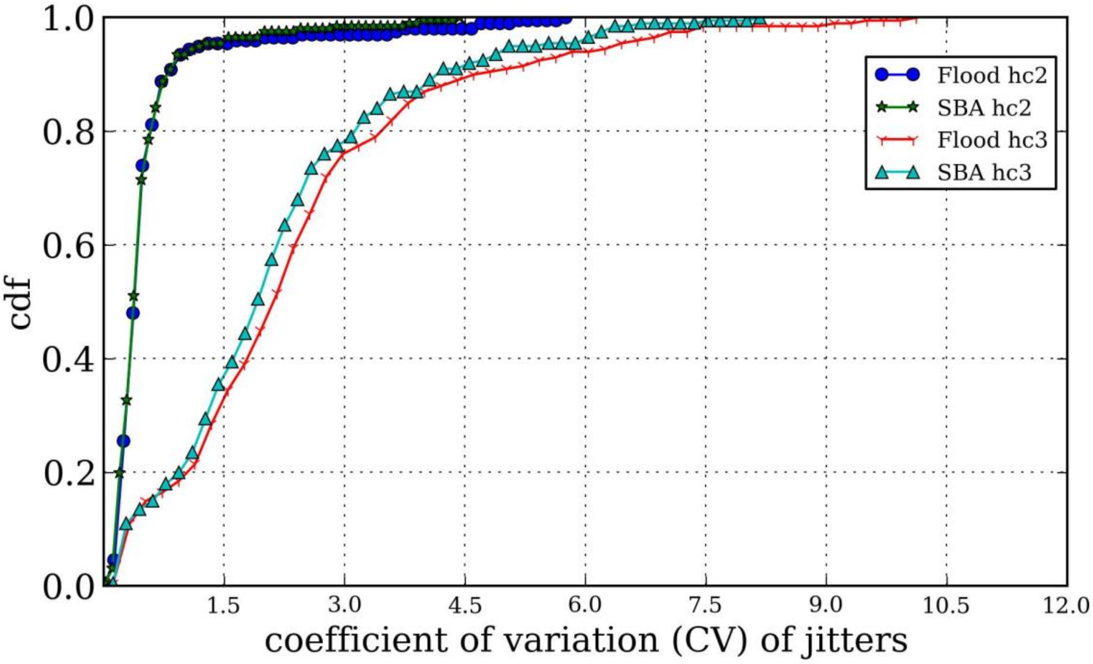


Figure 10.3: CDF of CVs for hop counts 2 and 3, range 200m

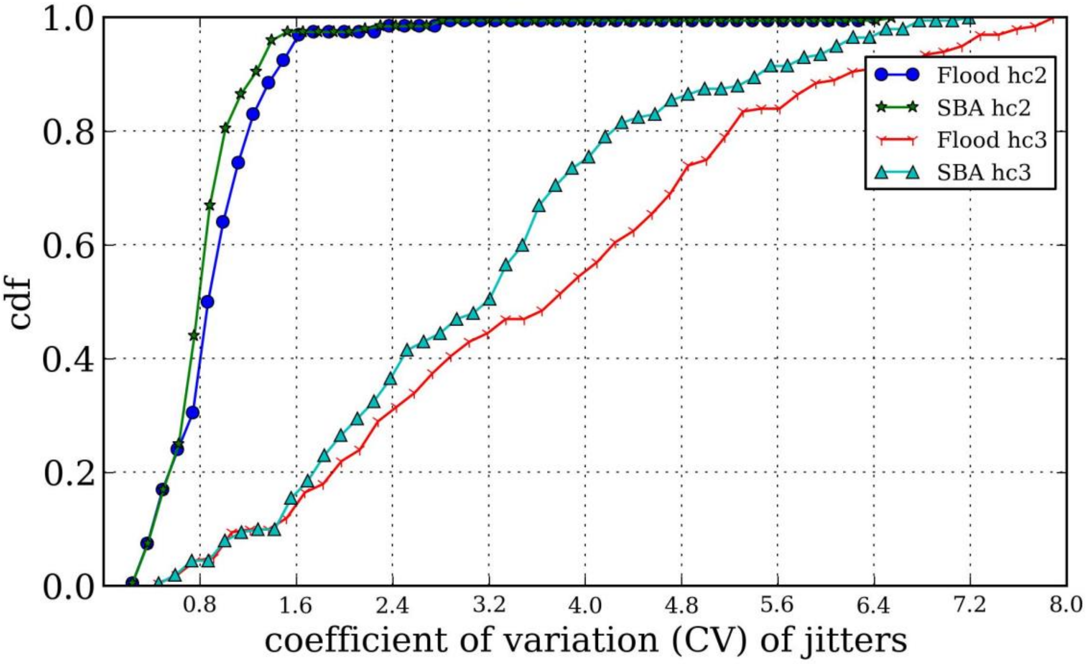


Figure 10.4: CDF of CVs for hop counts 2 and 3, range 600m

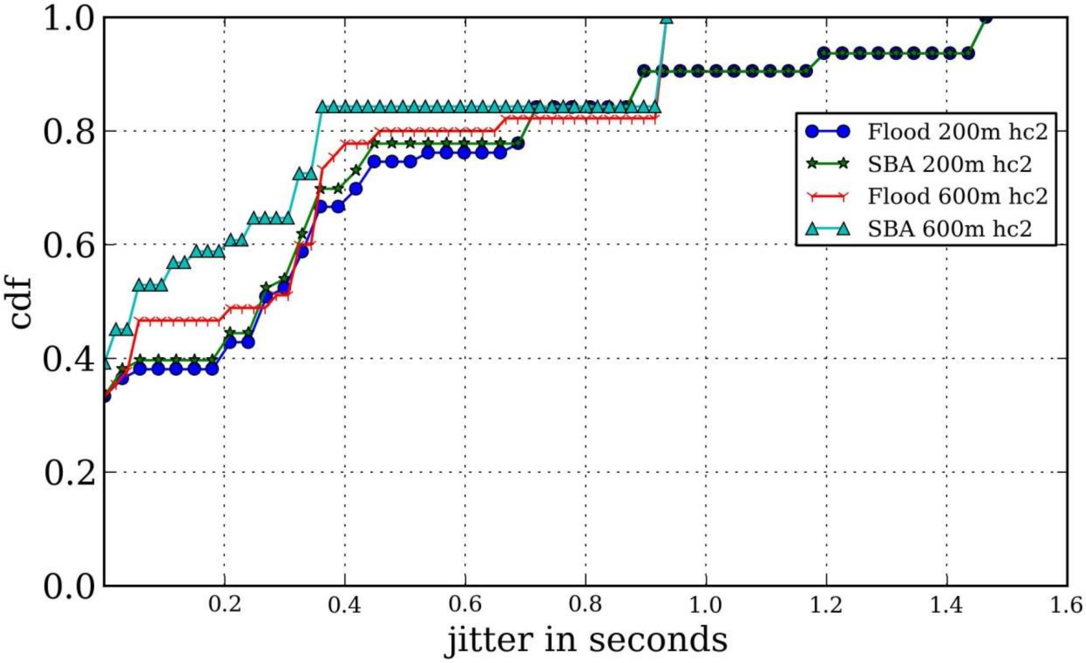


Figure 10.5: CDF of Jitters for Individual Packets for hop count 2

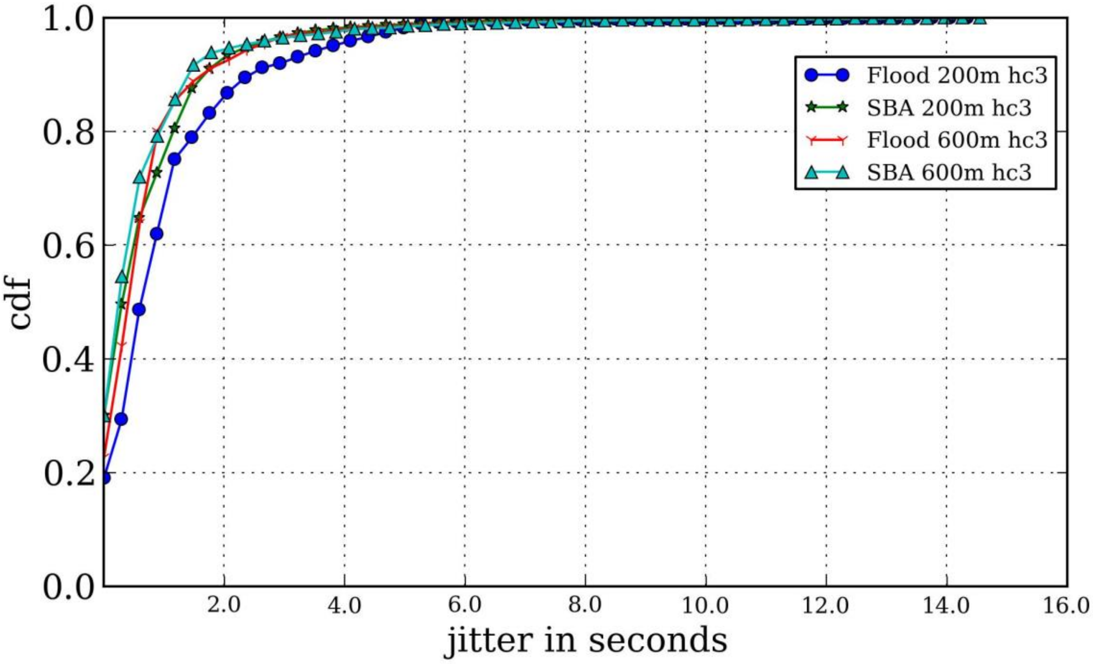


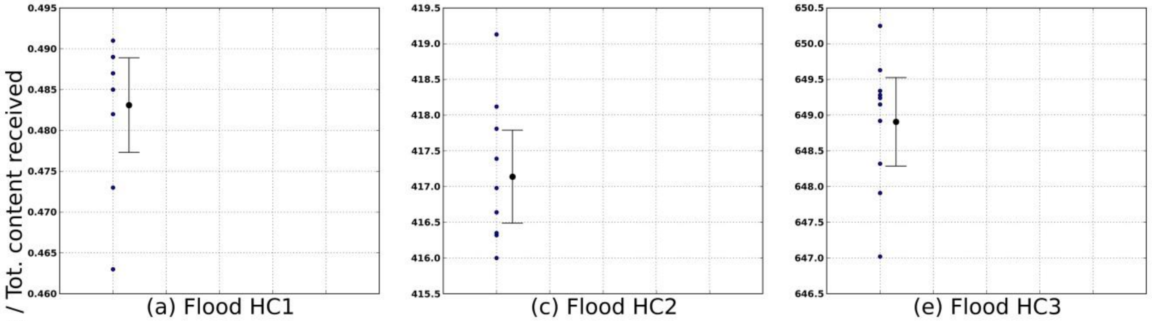
Figure 10.6: CDF of Jitters for Individual Packets for hop count 3

11 RELATION BETWEEN TOTAL TRANSMISSIONS AND NUMBER OF PACKETS RECEIVED BY THE GROUP MEMBERS (NODES)

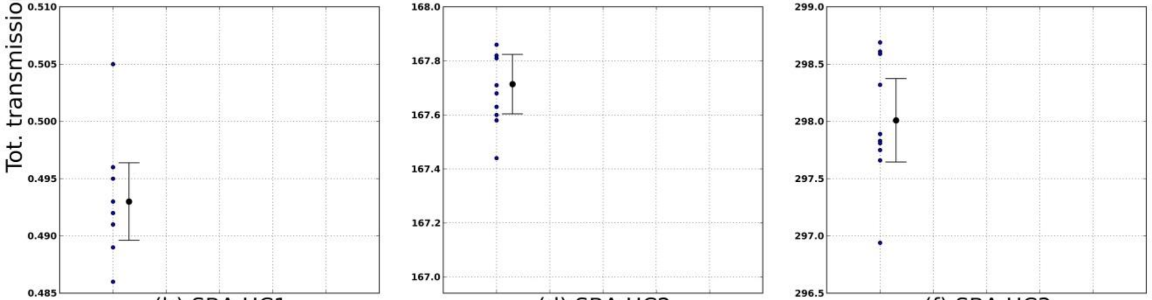
Finally, we calculate the ratio between the total transmissions, and the number of packets received by varying the broadcast scheme and their respective hop counts. We use these calculations to compare the performances of Flood and SBA broadcast techniques, and derive a conclusion for SBA's failure in high speed VANET scenario.

Figure 11.1 shows the confidence interval plots for the ratio between total transmissions and number of packets received by the nodes involved in the group talk. We notice that the 90% confidence intervals for SBA lie between 0.490 and 0.497 for hop count 1, 167.6 and 167.85 for hop count 2, 297.75 and 298.3 for hop count 3, which are significantly lower than the confidence interval values of Flood broadcast. It was interesting to note that though SBA significantly reduces the total transmission count, the total content received by the group members turned out to be significantly lower than Flood broadcast protocol.

To investigate the poor content delivery ratio of SBA broadcast, we analyzed the NS2 movement traces of our experiments. We found that the nodes frequently made incorrect decisions in forwarding the data packet. In many situations the nodes did not forward the packet to the intermediate nodes that serve as forwarders. This is primarily due to the incorrect information of nodes in the broadcast cover set that is maintained by every node in the simulation. Due to high mobility of vehicles the nodes do not contain precise information about their local neighborhood. The nodes assume that that all the nodes in its neighborhood have received the broadcasted packet. However, this is not true since the node’s neighborhood information is imprecise. This is further hastened by new vehicles that are introduced at regular intervals, which constantly travel at high speeds (65mph). Therefore, in our case SBA algorithm's [11] ability to reduce rebroadcasts has resulted in a negative impact on content delivery.



(a) Flood HC1 (c) Flood HC2 (e) Flood HC3



(b) SBA HC1 (d) SBA HC2 (f) SBA HC3

Figure 11.1: Total transmissions divided by the total content received for Flood and SBA broadcast schemes

12 FUTURE SCOPE

So far we have analyzed the feasibility of voice chat in vehicle caravanning. From our exhaustive set of results, it is quite clear that the group voice chat can serve as a potential infotainment application. The voice chat performs very well under Flood broadcast algorithm, and outperforms the voice chat using SBA broadcast. Apart from voice chat application, there are plenty of infotainment based applications that can prove quite beneficial in enhancing inter vehicle communication technology. For example, multiplayer games in vehicles that support on-the-go gaming service are witnessing a huge demand amongst young passengers. Nowadays, we have many multiplayer games specially designed for smart phones; some involve low graphics while some with heavy graphics. Implementing multiplayer games for vehicles can be quite challenging since games demand higher bandwidth and low latency. A multiplayer game can be initiated with any person who is not driving. People in other vehicles (travelling within a certain range of vehicles) can join the game that is initiated by the host vehicle. It would be quite interesting to know delay characteristics of such gaming applications. The other common application is the text chat and content sharing. These applications are one the most primitive, yet essential forms of infotainment. Since we explored the possibility of implementing voice chat in inter vehicle communication, the implementation of such text chats and content sharing are very much within reach. Text chat does not require higher bandwidths and strict delay requirements. Broadcasting road side advertisement is an effective form of advertising technique. For example, road side restaurants can broadcast their menus and current deals to vehicles travelling within the rage of the restaurant. In this way, the people can choose the best deals and meal plans to suit their needs. This form of communication is termed as V2I (Vehicle to infrastructure) communication. The voice chat can be extended in the form of real time video chat application; however, this could demand very high bandwidth requirement.

Hence we can see innumerable ways of extending the infotainment applications in vehicle. In order to make these applications practical, the NS2 and the VISSIM simulator can serve as excellent tools to perform the initial simulations. From this thesis we saw that the VISSIM is able to simulate realistic freeways with varying traffic densities. By using such traces in NS2, it possible to get results that are closer to real-world experiments.

13 CONCLUSION

In this thesis we have explored the viability of voice chat for vehicular infotainment. The key contributions of this thesis can be summarized as follows:

- Developing a realistic voice chat model for inter-vehicle communication resembling a casual conversation involving 3-5 people.
- Integrating the SBA broadcast algorithm into the NS2 VANET module.
- Generating realistic mobility trace depicting the I-75S freeway using VISSIM microscopic simulator.
- Performing extensive simulations to compare the performance of the voice chat model in Flood and SBA broadcast scenarios.

From our experimental results, we showed that the voice chat application is very much plausible in high speed VANET environments. In Section 5.1 we showed that the content delivery percentage of the voice packets was impressive for hop counts 2 and 3. We showed that the increase in transmission range results in higher percentage of content delivery.

In Section 8 we presented the results on the fairness of our voice chat algorithm; we found the maximum talk spurt duration to be less than 10 seconds with most of the talk spurts being 4 seconds long. Furthermore, we saw that none of the nodes dominated the talk duration. The maximum increase in the fairness value for both Flood and SBA broadcast were 2 steps (each step being 2 units). For a group size of 5 the transitions in fairness values were quite frequent and some of the nodes did not reach their minimum fairness value, but with a group size of 3 we found a steady decrease in the fairness value until all the nodes reached their minimum fairness value of 0. This clearly proved that the change in the group size results in the change in talking behavior. We also increased the node densities to find out its effect over talking behavior of the groups. The increase in node density resulted in frequent talk spurts, which was shown as dense peaks in Figure 7.7.

Finally we observed the efficiency of Flood broadcast over SBA broadcast algorithm. The content delivery ratio of Flood was significantly better than SBA irrespective of the increase in transmission range for SBA broadcast; however, SBA produced a lower end-to-end delay between broadcasting node and its group members. The SBA also has a lower transmission cost per packet received when compared to Flood broadcast. Therefore, both the protocols have their merits and demerits, and this result was one of the most interesting outcomes of this Thesis. In Section 10 we concluded by stating that the SBA’s poor content delivery is mainly due to incorrect neighborhood information, which is caused by high mobility and rapid injection of nodes into the VANET scenario.

REFERENCES

1. VISSIM, *http://www.vissim.de/*.
2. Amoroso, A., et al. *A Simulative Evaluation of V2V Algorithms for Road Safety and In-Car Entertainment*. in *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*. 2011. IEEE.
3. Tanabe, H., Y. Asano, and M. Satomura. *Evaluation of Connectivity Quality for Vehicle-to-Vehicle Communication System*. in *17th ITS World Congress*. 2010.
4. Shin, K.E., H.K. Choi, and J. Jeong. *A practical security framework for a VANET-based entertainment service*. in *Proceedings of the 4th ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*. 2009. ACM.
5. Han, L., et al. *Ad-hoc voice-based group communication*. in *Pervasive Computing and Communications (PerCom), 2010 IEEE International Conference on*. 2010. IEEE.
6. Kunz, T. *Multicasting in mobile ad-hoc networks: achieving high packet delivery ratios*. in *Proceedings of the 2003 conference of the Centre for Advanced Studies on Collaborative research*. 2003. IBM Press.
7. Rao, J. and T. Giuli. *Evaluating vehicular ad hoc networks for group applications*. in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*. 2011. IEEE.
8. Papp, G. and C. GauthierDickey. *Characterizing multiparty voice communication for multiplayer games*. in *ACM SIGMETRICS Performance Evaluation Review*. 2008. ACM.

9. Ho, C., et al. *Flooding for reliable multicast in multi-hop ad hoc networks*. in *Proceedings of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications*. 1999. ACM.

10. Ni, S.Y., et al. *The broadcast storm problem in a mobile ad hoc network*. in *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*. 1999. ACM.

11. Peng, W. and X.C. Lu. *On the reduction of broadcast redundancy in mobile ad hoc networks*. in *Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing*. 2000. IEEE Press.

12. Williams, B. and T. Camp. *Comparison of broadcasting techniques for mobile ad hoc networks*. in *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*. 2002. ACM.

13. Tonguz, O.K. and M. Boban, *Multiplayer games over vehicular ad hoc networks: A new application.* Ad Hoc Networks, 2010. **8**(5): p. 531-543.

ABSTRACT

PERFORMANCE EVALUATION OF VOICE CHAT IN VEHICULAR AD-HOC NETWORKS

by

VINEETH RAKESH MOHAN

December 2013

Advisor: Dr. Hongwei Zhang

Major: Computer Science

Degree: Master of Science

Inter vehicle communication has emerged as an important area of research. With a rapid evolution of social networks, people are constantly looking for social interactions in all types of mobile environment. In this Thesis, we propose a voice chat model for Vehicle-to-Vehicle (V2V) communication that mimics a real-world group talk scenario, and measure its performance using Flood and Scalable broadcast (SBA) protocols. To evaluate the performance of the voice chat, we use different parameters such as group size, network density, transmission range and hop counts to show that our voice chat application is highly feasible in VANET environment. Furthermore, we perform a thorough comparison of Flood and SBA broadcast protocols throughout our simulation. Contrary to the performance of SBA in low speed ad-hoc networks, we show that the Flood broadcast algorithm has better content delivery than SBA in all scenarios that we tested. We implement our model using the NS2 network simulator using a realistic vehicular trace that depicts the movement of vehicles in the I-75S freeway from Detroit to Toledo.