B.Sc. in Computer Science and Engineering Thesis

Simulation Study of Vehicular Mobility in City Streets

Submitted by

Md. Abdullah Al-Alamin
201205084

Supervised by

Dr. Md. Shohrab Hossain

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology
Dhaka, Bangladesh

September 2017

CANDIDATES' DECLARATION

This is to certify that the work presented in this thesis, titled, "Simulation Study of Vehicular Mobility in City Streets", is the outcome of the investigation and research carried out by us under the supervision of Dr. Md. Shohrab Hossain.

It is also declared that neither this thesis nor any part thereof has been submitted anywhere else for the award of any degree, diploma or other qualifications.

Md. Abdullah Al-Alamin
201205084

CERTIFICATION

This thesis titled, **“Simulation Study of Vehicular Mobility in City Streets”**, submitted by the group as mentioned below has been accepted as satisfactory in partial fulfillment of the requirements for the degree B.Sc. in Computer Science and Engineering in September 2017.

Group Members:

Md. Abdullah Al-Alamin

Supervisor:

Dr. Md. Shohrab Hossain
Associate Professor
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology

ACKNOWLEDGEMENT

I am thankful to Almighty Allah for His blessings for the successful completion of my thesis. I would like to thank my thesis advisor Dr. Md. Shohrab Hossain at Bangladesh University Of Engineering And Technology (BUET). The door to Sir's office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently steered me in the right the direction whenever he thought I needed it.

I would like to thank the authors of a previous work on city section mobility model [1]. This thesis work is the simulation study based on theoretical model proposed in [1].

Finally, I must express my very profound gratitude to my parents and to my family for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Md. Abdullah Al Alamin

Finally,

Dhaka
September 2017

Md. Abdullah Al-Alamin

Contents

List of Figures

List of Tables

ABSTRACT

Performance evaluation of vehicular network protocols require knowledge of different stochastic properties of the mobility model used in the evaluation. In this thesis work, the stochastic properties of vehicular mobility models in city [1] has been modified and validated by popular simulator VANETSim which run simulation on real street map data. The developed model can be used as a generic framework for comprehensive analysis of other mobility models.

Chapter 1

Introduction

Vehicular communication has become an essential technology for improved safely in city streets and highways. Mobility models that can mimic the movement pattern of vehicles are important building blocks for simulation-based studies for vehicular ad hoc networks (VANET) [1–5] and mobile networks [1, 6]; these models can help in testing and evaluating protocols related to vehicular mobile networks. Mobility should be modeled in a realistic manner to ensure match between simulation results and data from real-world deployment of vehicular mobile networks.

Vehicular mobility models [1,4] can be grouped into four categories: the first is synthetic model that is based on mathematical models while others are survey-based models, trace-based models and simulator-based models. Among these four variants, we are interested in synthetic models since mathematical models can help in obtaining the general trend and behavior of the mobility model. Moreover, sensitivity of certain system parameters can be easily analyzed in mathematical models which might not be possible for other types of mobility models.

In real life scenarios, while moving along city streets, vehicles do not have the freedom to travel freely along any arbitrary direction with a random speed. There are buildings, rivers, trees, etc, and the vehicular mobility environment is constrained by city streets. Moreover, vehicles must follow traffic regulations, such as, stop lights, speed limits, etc. Therefore, many widely used random (synthetic) mobility models [2, 3, 7–11], for example, random waypoint and random direction models are not suitable at all to represent vehicular movement.
In this paper [1] developed a mathematical model to mimic the movement pattern of vehicles roaming in a city, and termed it as *Vehicular Mobility model in City Streets (VMoCS)*. This model [1] takes into account the city street map and street constraints (such as, stop lights, speed limits, acceleration, deceleration, etc.) as expected in city streets. In this model [1], a vehicle travels from a random initial point (at crossroad) to another destination point, and such a movement is termed as an *epoch*. It [1] has derived several stochastic properties of the VMoCS model. To be specific, it [1] has derived closed form expressions for the distance traveled by a vehicle during each cycle, variance of epoch length, mean epoch time, number of

subnet crossing per epoch and the duration a vehicle resides under an access network (subnet).

Several works on vehicular mobility models, both analytical [1, 3, 7, 12–20]and simulation-based [21–26] have been reported in the literature.

1.1 Motivation

The stochastic properties of mobility model such as epoch length, variance of epoch length, expected epoch time, expected number of subnet crossing, and subnet residence time are very crucial for vehicular network protocols since they give an estimate of the amount of signaling required for mobile networks moving around the city. Without the proper knowledge of these properties and how they are affected by related system parameters, such vehicular mobility models cannot be accurately utilized by vehicular simulation tools that are needed for the performance evaluation of solutions proposed for vehicular mobile networks.
However, in the previous work [7, 12–20] the analytical models have not derived expressions for various stochastic properties (such as, mean epoch length, epoch time, and subnet residence time) that are very crucial for the utilization and applicability of any mobility model.
On the other hand, as this paper [1] deals with synthetic models, simulation-based models [21–26] are not the focus of this paper. Moreover, simulation studies cannot always be tested with a large range of parameter values due to resource limitation and may sometime fail to model the general trend of the problem.

1.2 Objective

Analytical models to represent general scenarios vehicular mobility provide better insights into the behavior of the system being analyzed. My *objective* in this paper is to develop an analytical model [1] for vehicular mobility pattern. This mobility model [1] considers city street constraints such as speed limit, one way, two way road etc. Our objective is to obtain its key stochastic properties for deeper understanding of its behavior. My Objective is to validate the paper's [1] analytical data and improve its limitations 3.2

1.3 Contribution

Earlier works on city section mobility model [1] developed a mathematical model to mimic the movement pattern of vehicles roaming in a city, and termed it as *Vehicular Mobility model*

in City Streets (VMoCS). This model [1] takes into account the city street map and street constraints (such as, stop lights, speed limits, acceleration, deceleration, etc.) as expected in city streets. My *contribution* in this paper is
(i) validating the analytical model presented in [1] through simulations using popular VANET-Sim simulator,
(ii) Solving the limitations 3.3 of the analytical model [1] and derive a modification to the existing analytical model [1] .
(iii) Compare and validate the above two mobility models 5.

1.4 Outline

The rest of the paper is organized as follows. A literature review on various mobility models is presented in Section 2. In Section 3.1, the VMoCS [1] mobility model is explained and its various stochastic properties are derived and analyzed. Then 3.3 tries to solve the limitations of Section 3.1. Section 4 describes the simulation setup and section 5 describes the result of the simulation. Finally, Section 6 concludes the paper.

Chapter 2

Literature Review

In this section, existing works related to mobility models are reviewed and then grouped into three categories: non-vehicular mobility models, analytical vehicular models and simulation-based vehicular models.

2.1 Classical random mobility models

A number of research works on random node mobility models have been reported in the literature. Bettstetter et al. [7, 8] analyzed the stochastic properties of the the most popular random mobility model, namely random waypoint model. Chiang et al. [1,7,9] proposed a 2-D random-walk model to analyze the cell crossing rate and dwell time of mobile device and applied for square and hexagon shaped cells. The analytical results have been validated by simulations. Ali et al. [10, 20] have used the different random mobility models for the analytical performance evaluation of media access algorithms of mobile voice users. These classical random mobility models [7–11, 20] are *not suitable for vehicular mobility modeling* since vehicular movement is always restricted by different constraints, such as, street map, stop lights, speed limits, etc, which have not been taken into account in the above studies.

2.2 Analytical vehicular mobility models

A number of analytical street mobility models exist in the literature. A great work [1] on vehicular mobility model has been done. This thesis mainly works on the extension of this paper [1]. A detailed survey on the available vehicular mobility models can be found in [4]. Bratanov et al. [1, 12] present a street mobility modeling for vehicle-borne terminals. They derive probability density functions for node movement direction, velocity, and street segment length, validated

by a street pattern tracing tool and via measurements.
Momen et al. [15] propose a stochastic vehicle mobility model that considers user mobility direction and velocity, described by discrete state Gauss-Markov technique. Chung et al. [18] propose a random (synthetic) vector mobility model for vehicular movement and claim that it can generate stochastically accurate mobility traces without requiring any map, street, signaling light information.
However, these works [12–18, 20] have *not derived expressions for stochastic properties*, such as, mean epoch length, epoch time, subnet residence time, etc that are strongly related to the metrics used for performance evaluation of vehicular network protocols and therefore, are crucial factors for the applicability of a mobility model.

2.3 Simulation-based vehicular mobility models

A number of simulation-based approach for street mobility modeling have been reported in the literature. Martinez et al. [1, 21] developed a simulation tool which is a mobility pattern generator to be used with the ns-2 simulator.
Saha et al. [1, 22] analyze a realistic street mobility model using a real street map; their tool generates scenarios in a format that is compatible with ns-2 format. Jardosh et al. [23] propose a mobility model that enables the inclusion of obstacles to restrict node movement and wireless transmissions in ad hoc network simulations.
Haerri et al. [19, 24] present VanetMobiSim, a generator of realistic vehicular movement traces for telecommunication networks simulators.
Huang et al. [26] investigates GPS data of thousands of taxis in an urban area to capture various characteristics of taxi mobility, which has been validated through extensive comparisons between the synthetic trace and the real trace.
However, these simulation-based models [3, 19, 21–26] are *not capable of capturing the generic characteristics* of the vehicular mobility pattern and hence cannot mimic mobility required for vehicular network protocols. Paper [1] has some limitation/restriction which I would try to improve in this thesis.

2.4 This work

This work is based on the work on VMoCS [1]. In the VMoCS model, the authors have considered many constraints present in city streets and incorporated realistic driving strategies while deriving closed form expressions for stochastic properties of the mobility model that are critical metrics for performance evaluation of any mobility and handover protocols. The correctness of the model through simulations whose parameters have been chosen from real street map data have been trid to be verified in this thesis work.

I found some limitations of the existing model [1] and tried to provide an extended analytical model. I also tried to verify this new mathematical model through VANETSim simulation. My work ie the extension of paper [1] therefore enables better utilization of the vehicular mobility model by characterizing and analyzing all its key properties.

Chapter 3

City Section Mobility Model

3.1 Original work

The analytical part of the City Section mobility model is derived in our previous work [1]. In the paper, the street map considered is represented by a grid [1] shown in Fig. 4.4. Each vehicle starts movement from a randomly picked crossroad. It then chooses a destination point (another crossroad in the street map). Movement to the destination involves (at most) one horizontal and one vertical movement. Upon reaching the destination crossroad, the vehicle again chooses another destination point and the process is repeated. Each such cycle is termed as an *epoch*.

Following are the assumptions of the city section mobility model:

- Starting and destination points are assumed to be crossroads.
- Each street and avenue have separate speed limits.
- Streets and avenues are parallel to axes.
- Stop light is present at every crossroad. A vehicle encounters the stop light at a crossroad with a probability.
- After each stop signal, each vehicle accelerate up to the speed limit of the street / avenue (in a fraction of the street segment) and moves with the speed unless a stop light is encountered.
- In case of encountering a stop light, the vehicle stops by decelerating in a fraction of street segment.

The street map shown in Fig. 4.4 [1] is a rectangular shaped area of dimension $a \times b$. Let there be N_s horizontal roads (*streets*) and N_a vertical roads (*avenues*) and streets be S_y distance apart and avenues be S_x distance apart. So number of streets and number of avenues are [1]

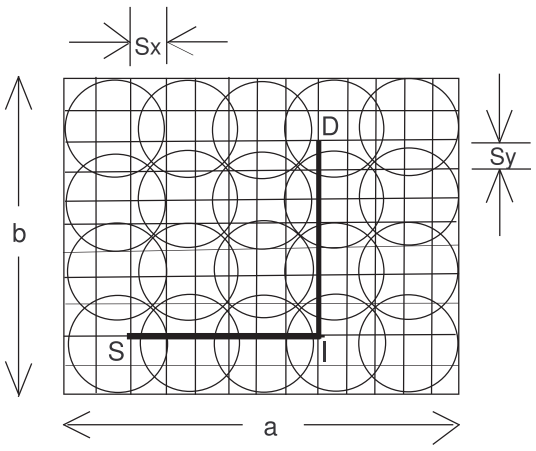


Figure 3.1: Road network in VMoCS mobility model. [1]

$$N_a = \frac{a}{S_x} + 1 \tag{3.1}$$

$$N_s = \frac{b}{S_y} + 1 \tag{3.2}$$

In the i-th epoch, the vehicle moves from point S^i to point D^i via intermediate point I^i, involving (at most) one horizontal and one vertical movement. So the movement pattern in each epoch of VMoCS model can be represented by (S^i, V_x^i,I^i), (I^i, V_y^i, D^i), where V_x^i and V_y^i are speed limits of the two streets. In an epoch where the starting point and destination point are on the same (horizontal or vertical) road segment, the intermediate point I^i coincides with D^i.

3.1.1 Epoch Length

In each epoch, the vehicle moves along (at most) two road segments. Let L_x^i and L_y^i be the lengths of those two road segments during i-th epoch. So $L_x^i = |I_x^i - S_x^i|$ and $L_y^i = \left|D_y^i - I_y^i\right|$, where P_x and P_y are the x and y-coordinates of the point P. So the total distance covered by the vehicle during i-th epoch is as follows [1]:

$$L^i = L_x^i + L_y^i \tag{3.3}$$

3.1.2 Expected Epoch Length

Let us first find out the expected length along the street (i.e., along horizontal direction). Let it be L_x.

we get [1]

$$E(L_x) = \frac{a(N_a + 1)}{3N_a} \tag{3.4}$$

Similarly, for movement along avenues, the expected value of L_y can be obtained as follows [1]:

$$E(L_y) = \frac{b(N_s + 1)}{3N_s} \tag{3.5}$$

Therefore, the expected epoch length can be obtained by adding Eqns. (3.4) and (3.5) as follows [1]:

$$E(L) = \frac{a(N_a + 1)}{3N_a} + \frac{b(N_s + 1)}{3N_s} \tag{3.6}$$

3.1.3 Variance of Epoch Length

The variance of epoch length implies how the epoch lengths vary about the mean epoch length.

Therefore, the variance of epoch length can be obtained from [1]

$$V(L) = \frac{S_x^2(N_a^2 - 1)}{18} + \frac{S_y^2(N_s^2 - 1)}{18} \tag{3.7}$$

for $S_x = S_y$ = 50 to 200 meters. For an $a \times a$ square shaped road network, $V(L) = 2.2444a$ km and 4.4667a km for $\{N_a, S_x\}$ = {100, 200 m} and {200, 200 m}, respectively. For an $a \times (a/2)$ rectangular road network, $V(L) = 1.4056a$ km and 2.7944a km for $\{N_a, S_x\}$ = {100, 200 m} and {200, 200 m}, respectively. Thus, it is found that the variance of VMoCS model increases with the increase of S_x (inter-road spacing) and N_a (number of avenues), since $V(L)$ is proportional to the square of S_x and N_a.

3.1.4 Epoch Time

The next step is to find out the total time required for a vehicle (mobile node) to complete an epoch. We have considered possible restrictions in the city streets, such as, stop lights, speed limits. It is assumed an average delay of τ sec at each crossroad with a probability of ϕ of encountering it, that is, the vehicle will encounter a stop light in a crossroad with a probability ϕ.

So the total delay regarding stoppages in crossroads can be estimated as follows [1]:

$$T_{xroad} = \phi\tau\left(\frac{E(L_x)}{S_x} + \frac{E(L_y)}{S_y}\right) \tag{3.8}$$

Hence, the total epoch time can be obtained from paper [1]

$$E(T) = T_{xroad} + E(T_x) + E(T_y) \tag{3.9}$$

Higher value of N_a means larger grid dimensions for fixed inter-road spacing (see Eqns. (3.1) [1] and (3.2)), resulting in higher epoch time. For a $a \times a$ square road network, $E(T)/a$ = 262.17 milisec/meter and 145.75 milisec/meter when $\{V^{max}$, $N_a\}$ = $\{$5 m/s, 200$\}$ and $\{V^{max}$, $N_a\}$ = $\{$ 25 m/s, 200$\}$, respectively. For a $a \times (a/2)$ rectangular road network, $E(T)/a$ = 178.96 milisec/meter and 91.02 milisec/meter when $\{V^{max}$, $N_a\}$ = $\{$5 m/s, 200$\}$ and $\{V^{max}$, $N_a\}$ = $\{$25 m/s, 200$\}$, respectively.

3.1.5 Number of Subnet Crossing

Let us consider that the road network of dimensions $a \times b$ (Fig. 4.4) [1] is covered by Access Points (AP); let there be n rows of APs and m APs in each row. In total, there will be mn APs to cover the rectangular area. Let the radio coverage area of each AP be a circular region of radius r and two successive APs overlap at a maximum length of l along its diameter.

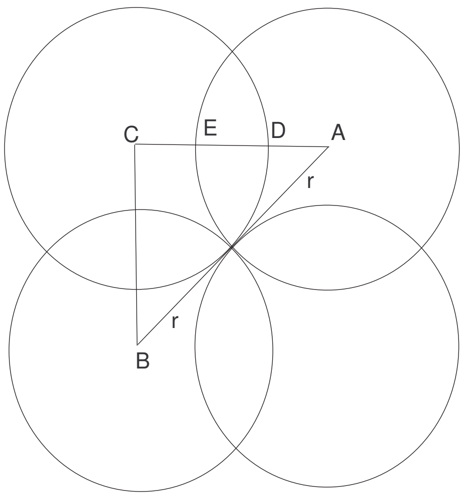


Figure 3.2: Subnet overlapping among the cells. [1]

Expected number of subnet crossings in an epoch for movement along the vertical direction is [1],

$$E(C_y) = \frac{n(n+1)K}{6{N_s}^2}(6N_s - 4nK + K + 3) \tag{3.10}$$

Therefore, the expected number of subnet crossing in an epoch can be obtained from [1]

$$E(C) = E(C_x) + E(C_y) \text{ 3.10} \tag{3.11}$$

expected number of subnet crossings increases with the increase in number of avenues (N_a) and streets (N_s). Higher values of N_a and N_s mean larger grid dimensions for fixed inter-road spacing (see Eqns. (3.1) and (3.2)), resulting in higher number of subnet crossings.

For a $a \times a$ square road network, $E(C)$ = 19.79 when K = 7, N_a = 200; and $E(C)$ = 39.85 when K = 3, N_a = 200. For a $a \times (a/2)$ rectangular road network, $E(C)$ = 15.04 when K = 7, N_a = 200; and $E(C)$ = 30.26 when K = 3, N_a = 200.

3.1.6 Subnet Residence Time

The subnet residence time is the time duration that the mobile node (vehicle) resides under a cell. This can be obtained by dividing the mean epoch time by expected number of subnet crossing as follows:

$$T_r = \frac{E(T)}{E(C)} \tag{3.12}$$

36 km $\times$ 36 km square-shaped and 36 km $\times$ 18 km rectangular-shaped road network. Here, we used $K = 5$. We find that the residence time decreases with the increase of speed limit as less time is required to cross the coverage area of an AP. Residence time also decreases with the increase in number of avenues (N_a) and streets (N_s). Higher values of N_a (N_s) result in increased $E(C)$ values, but the value of $E(T)$ is not affected much. As a result, the residence time decreases.

For a 36 km $\times$ 36 km square road network, T_r = 899.37 sec and 190.95 sec when $\{V^{max}$, $N_a\}$ = {10 m/s, 50} and {30 m/s, 200}, respectively. For a 36 km $\times$ 18 km rectangular road network, T_r = 767.18 sec and 154.28 sec when $\{V^{max}$, $N_a\}$ = {10 m/s, 50} and {30 m/s, 200}, respectively.

3.2 Limitations of City Street Mobility Model

Through the paper [1] finds some of the stocastic properties but it has some limitations in its derivation. Some of the limitations are stated below.

- Different speed variation at different streets was not considered.

- Chance of delays in crossroad was not considered in the paper [1].
- According to the paper [1] Streets and avenues are parallel to axes. But in reality they wont be exactly parallel
- A vehicle can have (at most) one horizontal and one vertical movement in epoch. But in real world a vehicle moves in zigzag path.

3.3 Modified Mobility Model

In this modification mainly two of the above drawback will tried to be solved.
i) Vehicles will move in zigzag path
ii) Streets and avenues wont always be parallel

We can solve the zigzag move problem by a little modification of the original analytical model. A zigzag path is essentially a summation of a few smaller simple epoch.

Modified

$$E(L)* = \sum_{i=1}^{n} E(L)$$

3.6 Here the E(L)* is the summation of n simple epochs. But the purpose of analytical equation we need to write E(L)* in a closed form.
One point to be consider in reality a vehicle wont choose the shortest path it will choose a which will take to to his destination in the shortest time with shortest distance. So the modified epoch length will be little more than the the derived epoch lenght [1].

E(L)* = E(L) + λ

Here λ is around 5%.

Now the epoch time will decrease because vehicle with try to reduce epoch time even in cost of epoch length.

E(T)* = E(T) 3.9 - ϕ

Here the value of ϕ is around 10%

Expected no of Subnet crossing will decrease because RSU will not be uniformly distributed. **Important road ie broad roads or busy roads** will have more RSU.

E(C)* = E(C) 3.11 + θ

Here θ value should be around 20%.

The result of the new modified model is given in section 5.2.

Chapter 4

Simulation Study

To validate the mathematical model, VANETSim [5] simulator has been used. Simulation environment, analysis of the results and a comparative discussion are presented in the following subsections.

4.1 VANETsim

VANETsim is a simulator which focuses on realistic vehicular movement and communication among vehicles. It has a useful graphical user interface and it has an experimentation environment which supports the user to set up or perform experiments. The VANET-Simulator is developed in Java 6 and can be run on most operating systems and hardware platforms. It is primarily used to verify security and privacy concepts in VANETs.

4.1.1 Its features

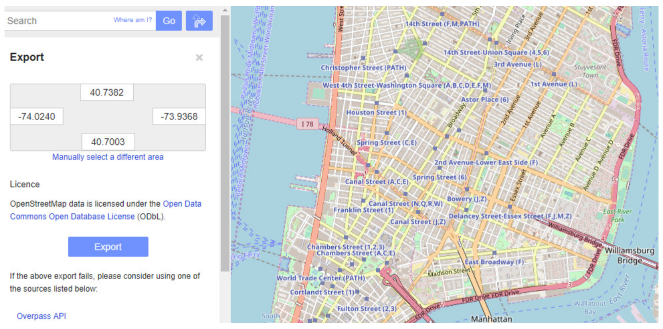
VANETsim's features are stated below.

- For routing An implementation of the A*-algorithm has been used. A* uses an heuristic to limit the distance calculations and thus results in a much faster routing compared with Dijkstra. It also always finds the optimal results like Dijkstra if the heuristic is correctly chosen (which it is in this implementation). An own data structure is used which is basically the official ¡code¿PriorityQueue¡/code¿ implementation but with unnecessary function calls and casts removed. This uses about 25% less cpu than the original PriorityQueue and about 40% less than a TreeSet. A basic ArrayList would take about 4x the performance.

- VANETsim simulator has real road constraints like one way streets, speed limit, checkpoint.
- The VANET-Simulator has an interface to import maps from the OpenStreetMap project. So the simulation of traffic on real road networks is supported. OpenStreetMap is built by a community of mappers that contribute and maintain data about roads, trails, cafes, railway stations, and much more, all over the world.
- Each vehicle is simulated individually and takes decisions on its own so that road traffic is simulated as realistic as possible.
- In VANETsim simulator vehicles communicate with each other and exchange information. So if there is a huge traffic jam at a crossing then other vehicles knows about this beforehand and choose another route.
- Streets have lots of realistic features. waypoints of streets are used for routing. Each street has its own steed limit. Streets can be unidirectional or bidirectional.
- A scenario is what saves the vehicles information and events. Scenarios are exported as XML and it contains information about the vehicle start point, end point, speed, wifi signal and others.
- It has classes to parse XML. Scenario and maps information are in XML format the simulator uses this parser to parse these information
- Streets have RSUs(A Road-Side-Unit to intercept WiFi signals). This is used to find when a vehicle moves from one network zone to another.
- There is an event class that handles all the events. Events can be blocking or unblocking streets and so on.
- Vehicles exchange information among themselves throgh message.A message which indicates some kind of traffic jam through assigning a penalty to the street on which the jam is.
- Master thread delegates the simulation processing to sub-threads and then calls a repaint on the drawing area
- Worker thread is meant to run parallel with multiple others to gain advantage of multiple CPUs. All simulation tasks are initiated from this class!
- Map has junction, traffic light, streets, waypoint, regions, node

4.2 My Contribution

- A simulation area was needed to run the simulation. So I researched and the = Manhattan streets match the analytical models requirement closely as it has relatively better street structure and street information is available from the OpenStreetMap. So I exported lower Manhattan (4 KM $\times$ 9 km) map from OpenStreetMap. 4.1.



OpenStreetMap data is licensed under the Open Data Commons Open Database License (ODbL).

If the above export fails, please consider using one of the sources listed below:

Figure 4.1: Exporting Map through OpenStreetMap

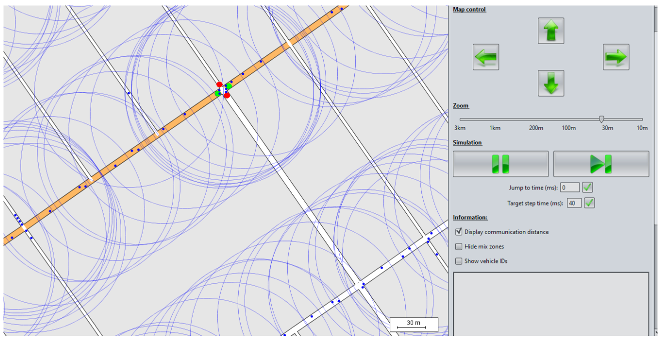
- Now scenarios are needed to perform simulation. A scenario is a collection of information about vehicles and vehicles' journey. So I Created different scenarios for the Simulation through VANETSim interface and stored those scenarios information in XML format.

- I needed to modify VANETsim's source code to log simulation data. VANETSim had its own logging features but i needed to write my custom log class A.1 to log simulation data which were useful to me. During the simulation when a vehicle reached destination vehicle's travel distance, travel time, wait time, no of subnet crossings etc were logged into the log file.

- VANETsim has 60+ classes to run the simulation. Among these classes some of the important classes that I modified or studied are VanetSimStart, Vehicle, Street, RSU, MAP, Node, XML Parser, Path, Message, Region, GUI event, A Star Algorithm, WayPoint, Scenario, Message etc.

- By default during the simulation the vehicles exchange messages 4.2. to communicate with each other to share information to know about the road information to choose optimal routes. I changed code so that the vehicles will share less information with streets traffic jam.

- For the purpose the thesis I needed to work on the VANETsim's vehicle class. I needed to add, modify some of the code to log vehicles' routing paths. When a vehicle enters a new street its entrance time were recorded and when the vehicle exited the street the amount of time the vehicle spent on that street were recorded. On the path when the vehicle came

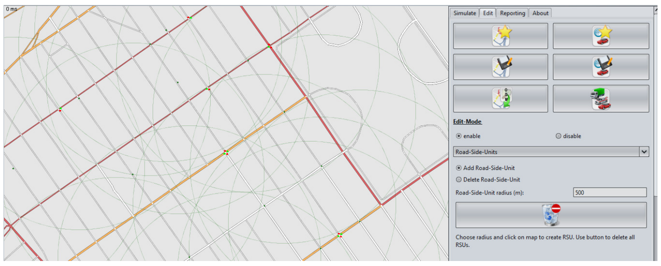


Jump to time (ms): 0
Target step time (ms): 40
[✓] Display communication distance
[ ] Hide mix zones
[ ] Show vehicle IDs

Figure 4.2: Vehicles within the radius are exchanging information among themselves

across a cross point i.e traffic light the amount of time spent on the traffic Signal were recorded.When a vehicle finished its epoch its total travel time, total length, total no of crossings were logged into a file to be analyzed later.

- Simulation was needed to run on different scenarios. Different Scenarios had different no of vehicles. The simulation time were also different on each run.
- The exported map from the OpenStreetMap did not have any Access Points(RSU). So I manually put Access Points in the Map 4.3.



[✓ enable] disable
Road-Side-Units
[✓ Add Road-Side-Unit] Delete Road-Side-Unit
Road-Side-Unit radius (m): 500
Choose radius and click on map to create RSU. Use button to delete all RSUs.

Figure 4.3: Add RSU to the map through VANETSim

- Through the exported map from OpenStreetMap were very realistic but it did not have all the information that i needed. The map region exported for the simulation through OpenStreetMap did not have all traffic signal information(Stop, One Way, Do Not Enter sign) in the map. So I had to manually put these traffic information in the map throught VANETsim's interface and thus I updated the exported map.
- I needed to modify the simulation's main Thread's code so that when simulation is terminated manually the current vehicles' information are correctly logged into the log file.
- Average amount of time a vehicle spent on a traffic signal was also logged A.2.
- VANETsim has a config file that stores simulation configuration. I also created my own config file to easily run, operate or log the simulation.

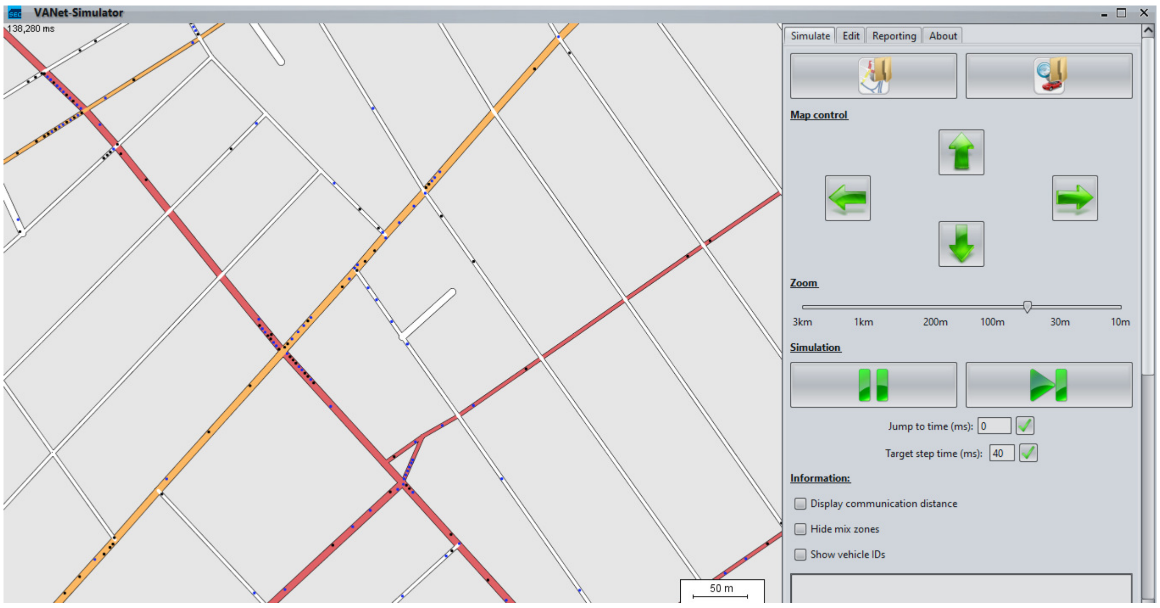
4.2.1 Problems Faced

- For the purpose of thesis bigger simulation area would have been better but map big region could not be exported from from OpenStreetMap could not be exported. Every time I tied to export a bigger area the OpenStreetMap failed. I exported lower Man Hat-tan area size (4 KM × 9 km) from OpenStreetMap
- During simulation a worker thread(created by the master Thread) was corrupted due my some of modification of code and this crashed the whole simulator.
- When I was trying to modify the code of the VANETSim I could not understand some of parts of the codes and so I emailed the author of VANETsim for help I did not get any reply. Finally this year in April the VANETsim project was closed by the author. So I was difficult to feedback from the author.
- At one point in vehicle class I checked the status of every vehicle on the map and updated some info. These unnecessary checkup was done frequently. So when the simulation ran for a long time with lots of vehicle the simulation froze up. Because it was doing unnecessary calculation for every vehicle and the time and memory it took for those calculation froze up the VANETSim.
- Before i used log every info of a vehicle into the log file when the vehicle finished its epoch ie reached its destination. So if a scenario had lots of vehicles in it and the simulation ran long time and lots of vehicles finished its epoch and were trying to write to the file. the simulation slowed down and performances were affected.

4.3 Simulation Setup

To estimate certain model parameters (such as, inter-road spacing, speed limit) for VMoCS model, we have considered the Manhattan road maps in the City of New York since it matches our assumed road network. It has been found the inter-street spacing in Manhattan is around 80-100 m and inter-avenue spacing is around 320-340 m. The speed limit is roughly 30 miles/hour. We have measured these parameters by analyzing Google Maps data. The signaling time cycle at the crossroads of New York city is between 45 and 120 seconds.

Based on the above observation, we have chosen our simulation parameter values. The default values of parameters used for simulations are listed in Table below. We used a rectangular shaped road network with a dimension of $4km \times 9$ km. We assumed the inter-street spacing is 90m and inter-avenue spacings to be the 200m. Therefore, number of avenues in the road network is 20 and number of streets is 100. The speed limit is assumed to be 15 meter/sec (which is around 33.55 miles /hr, reasonable for city streets). Average Manhattan traffic speeds,



Jump to time (ms): 0

Target step time (ms): 40

Figure 4.4: Vanet Simulation

from 9.35 mph to 8.51 mph(13.7 kph). The map has around 1700 traffic junction. Delay in each crossroad is assumed to be 50 sec to 120 sec and there is a 40% chance of encountering stop light in a crossroad. The acceleration portion of the street segment is assumed to be 10% of that segment. Similar is the case for deceleration phase.

4.4 Simulation Description

In our VMoCS model, we have considered many constraints present in city streets and incorporated realistic driving strategies while deriving closed form expressions for stochastic properties of the mobility model that are critical metrics for performance evaluation of any mobility and handover protocols. We have also verified the correctness of the model through simulations whose parameters have been chosen from real street map data. Real road scenario like traffic jam, one way street, check point are considered in the simulation

To the best of our knowledge, there exists *no such previous work* that quantitatively analyzed stochastic properties of a street mobility model. This work *differs* from the previous works in this respect and *presents a complete and detailed mathematical analysis* for vehicular mobility in city streets. Our work therefore enables better utilization of the vehicular mobility model by characterizing and analyzing all its key properties.

In each simulation run there were used different scenarios. Each scenario had had its own settings. In the scenarios the number of vehicles varied from 200 to 2000 at a time. We ran the simulation for 6-12 hours and took measurement of like 50 Thousand to 150 Thousand epoch.

Table 4.1: Values of parameters used in the simulation.

Simulation Parameters	**Values**
Simulation area	4 km $\times$ 9 km
Inter-road spacings (S_x)	200 meter
Inter-road spacings (S_y)	90 meter
Speed limit of each street / avenue	15 meter/sec
Number of Avenues (Streets)	101
Prob. of encountering stoplight at crossroad (ϕ)	0.4
Average delay in crossroad (τ)	60 sec
Acceleration / Deceleration portion of street segment (χ)	0.1
Number of MHs	20
Number of Epochs per MH per simulation	10
Wireless range	600 m
Wired link BW	10 Mbps
Wired link delay	1.8 ms
Wireless (802.11b) link BW	11 Mbps
Number of Access Points	1764
Number of Traffic Signal	1700

Then we calculated the parameter of E(L), VAR(L), E(T), E(C), E(Tr) We ran simulation on 10 different scenarios and the took the average of these parameters.

To estimate certain model parameters (such as, inter-road spacing, speed limit) for VMoCS model, we have considered the Manhattan road maps in the City of New York since it matches our assumed road network. It has been found the inter-street spacing in Manhattan is around 80-100 m and inter-avenue spacing is around 320-340 m. The speed limit is roughly 30 miles/hour. We have measured these parameters by analyzing *Google Maps* [27] data. The signaling time cycle at the crossroads of New York city is between 45 and 120 seconds [28].

Based on the above observation, we have chosen our simulation parameter values. The default values of parameters used for simulations are listed in Table 4.1. We used a square shaped road network with a dimension of 4 km $\times$ 9 km. We assumed the inter-street and inter-avenue spacings to be the same (240 m). Therefore, number of avenues in the road network is 151. Similar is the case for number of streets. The speed limit is assumed to be 15 meter/sec (which is around 33.55 miles /hr, reasonable for city streets). Delay in each crossroad is assumed to be 50 sec (maximum value) and there is a 40% chance of encountering stop light in a crossroad. The acceleration portion of the street segment is assumed to be 10% of that segment. Similar is the case for deceleration phase. Each access point's transmission range is 600 m. IEEE 802.11b standard is used for wireless communications. The wireless link bandwidth is 11 Mbps and wired link bandwidth is 10 Mbps.

Chapter 5

Simulation Result

5.1 Simulation Results

The results obtained from the analysis of the simulation traces are presented in this subsection. We have measured mean epoch length, epoch time, number of subnet crossings, residence times of each simulation run.

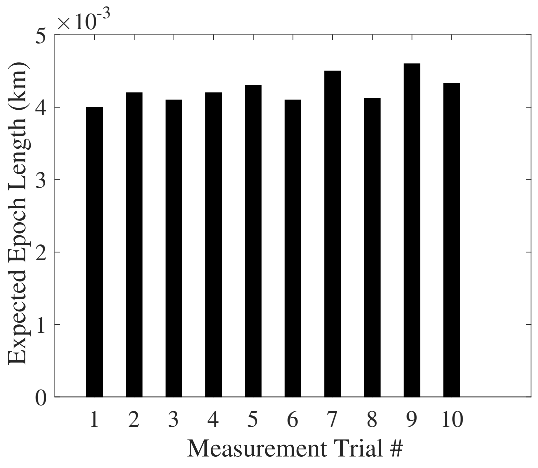


Figure 5.1: Expected epoch lengths for different simulation trials.

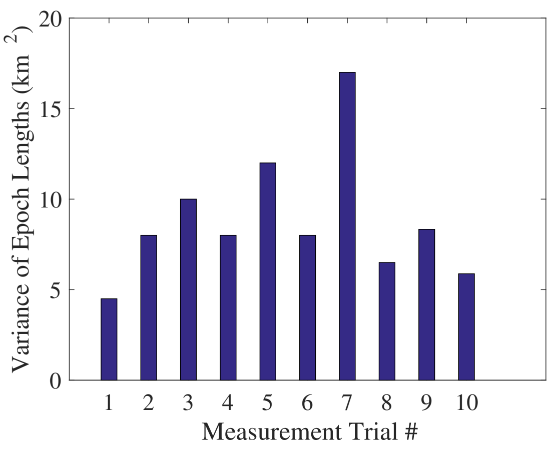


Figure 5.2: Variance of epoch lengths for different simulation trials.

Expected Epoch Length

Fig. 5.1 shows the bar chart of epoch lengths for each simulation run (having 200 epochs iterated in each run). There are 40 such simulation results. The average epoch length of all the simulation runs is 24.2113 km which is shown using the red straight line in Fig. 5.1. Variation among the epoch length values reflect the randomness of the mobility model.

According to the theoretical model presented in Section 3.1, the expected epoch length for the road network is given by Eqn. (3.6). Hence, $E(L) = E(L_x) + E(L_y)$ = 4 * (21 + 1) / (3*21) + 9 * (101+1) / (3 * 101) = 4.42 km. This value is close to the average epoch length obtained from simulation results with an accuracy of ? %.

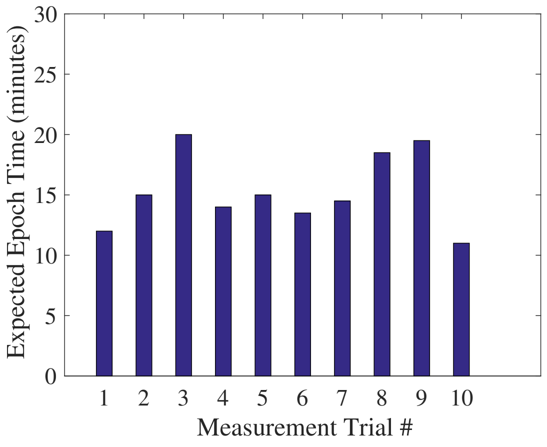


Figure 5.3: Expected epoch times for different simulation trials.

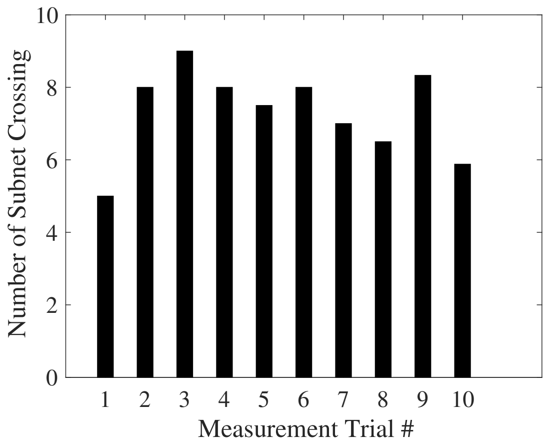


Figure 5.4: Bar chart for expected number of subnet crossing for different simulation trials.

Variance of Epoch Length

During each simulation run, there were 50-150 thousand epochs. We recorded the epoch lengths in each trial and computed the mean epoch length of each trial. Using the mean value, we have computed the variance of epoch lengths in each trial as shown in Fig. 5.2. The average variance of all the 10 simulation is 9.71 km^2. Hence, the standard deviation for epoch lengths is 3.11 km.

Using the analytical model, the variance of epoch lengths (for S_x = 200 meter, S_y = 90 meter, and N_a = 21 N_s = 101) can be obtained using Eqn. (3.7) and it is 5.56 km^2. Therefore, the standard deviation for a 4 km $\times$ 9 km square shaped road network is 3.11 km. Thus, the theoretical value matches the simulation one (accuracy of ? %).

Epoch time

Epoch time includes delay in the crossroads and time to travel horizontal and vertical road segment in an epoch. The theoretical value for delay in crossroads can be obtained using Eqn. (3.8) as T_{xroad} = 0.3 * 60 * (1390 / 200 + 3029.7 / 90) = 2013.2 sec. In addition, time required to travel horizontal and vertical street movement can be computed by Eqns. (**??**) and (**??**) which are T_x = 811.62 sec and T_y = 811.62 sec. Therefore, the calculated epoch time (from the analytical model) is $E(T)$ = 2013.2 + 811.62 + 811.62 = 3636.44 sec = 17.24 min.

On the other hand, the distribution for epoch time obtained from simulations are shown in Fig. 5.3. The average value is 3.88 min which is shown using the red straight line. The simulation value of mean epoch time are close to the analytical one with an accuracy of ?%.

Expected number of subnet crossings

Using the analytical model, the expected number of subnet crossing can be obtained using Eqns. (**??**), (3.10), and (3.11). For the square shaped road network of 36 km $\times$ 36 km, number of streets (avenues) are 151, the value $K = a/(mS_x) = 3.57$. Using the Eqn. (**??**), we find that $E(C_x) = 14.74 = E(C_y)$ due to square road shaped road network. Hence, $E(C) = E(C_x) + E(C_y) = 5.82$.

From the simulation traces, the average number of subnet crossing in each trial is obtained and Fig. 5.4 shows the corresponding bar chart. The average value is 29.208 shown using the red straight line. This average value is close to the analytical value with an accuracy of 99.05%.

Table 5.1: Comparison between analytical and simulation results.

Results type	**E(L)**	**V(L)**	**E(T)**	**E(C)**	**T$_r$**
Analytical	4.42 km	5.56 km^2	17.24 min	5.82	177.66 sec
Simulation	4.28 km	9.71 km^2	13.88 min	3.75	123.12 sec

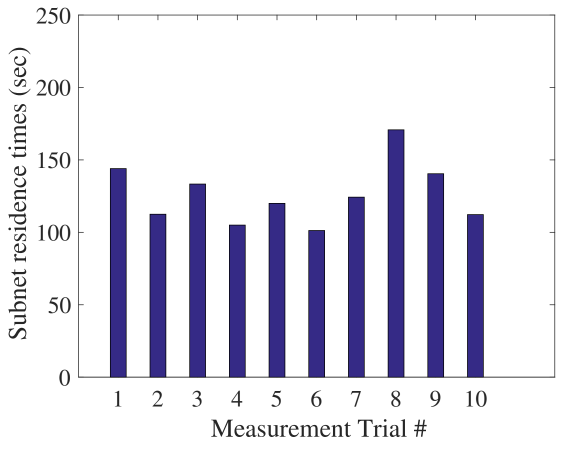


Figure 5.5: Bar chart for subnet residence times for different simulation trials.

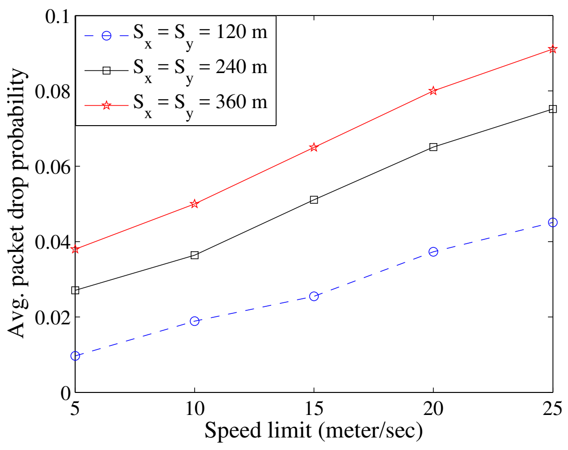


Figure 5.6: Average packet drop probability vs. speed limit for different inter-road spacing.

Subnet Residence Time

The subnet residence time can be computed using Eqn. (3.12) as T_r = 17.24 min / 5.82 = 177.66 sec. On the other hand, we show the subnet residence times for each simulation trial in Fig. 5.5. The distribution is almost flat and the average value of all simulation trial is 93.12 sec. Thus, we find that the simulation value matches the analytical one with ?% accuracy.

In Table 5.2, the results obtained from the mathematical model (Section 3.1) and the VANET-Sim simulation are presented for a 4 km $\times$ 9 km rectangular-shaped road network with inter-road spacing of 200 m and 90 m. The theoretical values of mean epoch length, variance of epoch length, epoch time, mean number of subnet crossing and subnet residence time matches the values obtained by simulation, thus validating our analytical model.

The objective of the paper is to derive several stochastic properties of realistic vehicular mobility model in city streets. These properties have been validated by VANETSim simulation results in Figs. 5.1-5.5.

5.2 Modified Result

The Modifed result based on the modifed model in section 3.3.

Table 5.2: Comparison between modified analytical and simulation results.

Results type	**E(L)**	**V(L)**	**E(T)**	**E(C)**	**T$_r$**
Modified Analytical	4.64 km	5.56 km^2	15.51 min	4.656	177.66 sec
Simulation	4.28 km	9.71 km^2	13.88 min	3.75	123.12 sec

So now we can see the modified model is giving better result.

Chapter 6

Conclusion

In this thesis, we have performed extensive simulation using VANETsim to validate our earlier work on city section mobility model that captures real driving strategies along with possible constraints in city streets, such as, stop lights at crossroad, speed limits, etc. We have performed a complete and detailed analysis of the model deriving the expressions for the following stochastic properties: expected epoch length, variance of epoch length, expected epoch time, expected number of subnet crossings and subnet residence time.
The derived stochastic properties have been validated by VANETSIM simulations whose parameters have been chosen from real street map data. Simulation results have been found to be in close agreement with the analytical model, having an accuracy of 83% for most of the stochastic properties.

In addition, this work can help in estimating the various performance metrics of mobility-enabled devices on-board vehicles moving in city streets.

References

[1] M. S. Hossain and M. Atiquzzaman, “Stochastic properties and application of city section mobility model,” *IEEE GLOBECOM*, 2009.

[2] S. Al-Sultan, M. M. Al-Doori, A. H. Al-Bayatti, and H. Zedan, “A comprehensive survey on vehicular ad hoc network,” *Journal of network and computer applications*, vol. 37, pp. 380–392, 2014.

[3] A. Pramanik, B. Choudhury, T. S. Choudhury, W. Arif, and J. Mehedi, “Simulative study of random waypoint mobility model for mobile ad hoc networks,” in *Communication Technologies (GCCT), 2015 Global Conference on*, pp. 112–116, IEEE, 2015.

[4] J. Harri, F. Filali, and C. Bonnet, “Mobility models for vehicular ad hoc networks: a survey and taxonomy,” *IEEE Communications Surveys and Tutorials*, vol. 11, pp. 19–41, Dec 2009.

[5] A. Tomandl, D. Herrmann, K.-P. Fuchs, H. Federrath, and F. Scheuer, “Vanetsim: An open source simulator for security and privacy concepts in vanets,” in *High Performance Computing & Simulation (HPCS), 2014 International Conference on*, pp. 543–550, IEEE, 2014.

[6] A. Shahriar, M. S. Hossain, and M. Atiquzzaman, “A cost analysis framework for NEMO prefix delegation-based schemes,” *IEEE Transactions on Mobile Computing*, vol. 11, pp. 1192–1206, Jul 2012.

[7] P. Nayak and P. Sinha, “Analysis of random way point and random walk mobility model for reactive routing protocols for manet using netsim simulator,” in *Artificial Intelligence, Modelling and Simulation (AIMS), 2015 3rd International Conference on*, pp. 427–432, IEEE, 2015.

[8] C. Bettstetter, H. Hartenstein, and X. Prez-Costa, “Stochastic properties of Random Waypoint mobility model,” *Wireless Networks*, vol. 10, pp. 555–567, Sep 2004.

[9] K.-H. Chiang and N. Shenoy, "A 2-d random-walk mobility model for location-management studies in wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 53, Mar 2004.

[10] T. Ali and M. Saquib, "Performance evaluation of wlan/cellular media access for mobile voice users under random mobility models," *IEEE Transactions on Wireless Communications*, vol. 10, pp. 3241–3255, Oct 2011.

[11] E. Zola and F. Barcelo-Arroyo, "Probability of handoff for users moving with the random waypoint mobility model," in *IEEE Conference on Local Computer Networks*, (Bonn, Germany), 2011.

[12] P. I. Bratanov and E. Bonek, "Mobility model of vehicle-borne terminals in urban cellular systems," *IEEE Transactions on Vehicular Technology*, vol. 52, pp. 947–952, Jul 2003.

[13] G. H. M. F. A. A. Javanmard and M. Hamdi, "Mobility modeling, spatial traffic distribution, and probability of connectivity for sparse and dense vehicular ad hoc networks," *IEEE Transactions on Vehicular Technology*, vol. 58, pp. 1998–2007, May 2009.

[14] K.-T. Chen, S.-L. Su, and R.-F. Chang, "Design and analysis of dynamic mobility tracking in wireless personal communication networks," *IEEE Transactions on Vehicular Technology*, vol. 51, May 2002.

[15] A. R. Momen and P. Azmi, "A stochastic vehicle mobility model with environmental condition adaptation capability," *Wireless Communications and Mobile Computing*, vol. 9, pp. 1070–1080, Aug 2009.

[16] A. Clementi, A. Monti, and R. Silvestri, "Modelling mobility: A discrete revolution," *Ad Hoc Networks*, vol. 9, pp. 998–1014, Aug 2011.

[17] K. A. Ali, M. Lalam, L. Moalic, and O. Baala, "V-mbmm: Vehicular mask-based mobility model," in *9th International Conference on Networks*, (Menuires, France), Apr 11-16, 2010.

[18] J. Chung and D. Go, "Stochastic vector mobility model for mobile and vehicular ad hoc network simulation," *IEEE Transactions on Mobile Computing (published online)*, Aug 2011.

[19] N. Akhtar, S. C. Ergen, and O. Ozkasap, "Vehicle mobility and communication channel models for realistic and efficient highway vanet simulation," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 1, pp. 248–262, 2015.

[20] M. Gorawski and K. Grochla, "Review of mobility models for performance evaluation of wireless networks," in *Man-Machine Interactions 3*, pp. 567–577, Springer, 2014.

[21] F. J. Martinez, J.-C. Cano, C. T. Calafate, and P. Manzoni, "CityMob: A mobility model pattern generator for VANETs," in *IEEE ICC Workshops*, (Beijing, China), May 2008.

[22] A. K. Saha and D. B. Johnson, "Modeling mobility for Vehicular Ad hoc Networks," in *ACM International Workshop on Vehicular Ad Hoc Networks (VANET)*, (Philadelphia, PA), Oct 2004.

[23] A. Jardosh, E. M. Belding-Royer, K. C. Almeroth, and S. Suri, "Towards realistic mobility models for mobile ad hoc networks," in *International Conference on Mobile Computing and Networking (MOBICOM)*, (San Diego, CA), Sep 14-19, 2003.

[24] J. Harri, F. Filali, C. Bonnet, and M. Fiore, "VanetMobiSim: Generating realistic mobility patterns for VANETs," in *ACM International Workshop on Vehicular Ad Hoc Networks (VANET)*, (Los Angeles, CA), Sep 29, 2006.

[25] J. Kim, V. Sridhara, and S. Bohacek, "Realistic mobility simulation of urban mesh networks," *Ad Hoc Networks*, vol. 7, pp. 411–430, Mar 2009.

[26] H. Huang, Y. Zhu, X. Li, M. Li, and M.-Y. Wu, "Meta: A mobility model of metropolitan taxis extracted from gps traces," in *IEEE Wireless Communications and Networking Conference*, (Sydney, Australia), Apr 18-21, 2010.

[27] "Google Maps." http://maps.google.com.

[28] "New york city: Department of transportation." http://www.nyc.gov/html/dot/html/faqs/faqs_signals.shtml.

Appendix A

Codes

A.1 Logging Code

This class is used to log vehicle data into log file…

```
package log.custom;

import java.io.BufferedWriter;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.io.UnsupportedEncodingException;
import java.io.Writer;
import java.lang.reflect.Array;
import java.util.ArrayList;

import vanetsim.map.Map;
import vanetsim.map.Region;
import vanetsim.scenario.Vehicle;

public class LogVehicleData {
        private Writer writer = null;
        private String outputFileName = "logs/log_output.txt" ;
        private static String fileWriterFileName = "logs/vehicle_street_l
        public static FileWriter fileWriter;

```

```
//      public LogVehicleData(){
//
//          try {
//                          writer = new BufferedWriter(new OutputStreamWrite
//                                      new FileOutputStream(outputFileName), "
//
//              } catch (UnsupportedEncodingException e) {
//                          // TODO Auto-generated catch block
//                          e.printStackTrace();
//              } catch (FileNotFoundException e) {
//                          // TODO Auto-generated catch block
//                          e.printStackTrace();
//              } catch (IOException e) {
//                          // TODO Auto-generated catch block
//                          e.printStackTrace();
//              }
//      }


        /**
         * This Method is a helper method to get all the vehicles active
         * Map has all the regions and regions has all the vehicles.
         *
         * @returns an arraylist of Vehicles
         */

//      public ArrayList<Vehicle> getAllVehicle(){
//
//              ArrayList<Vehicle> allVehicles = new ArrayList<>();
//
//
//              Region[][] regions = Map.getInstance().getRegions();
//              Vehicle[] vehicles;
//              Vehicle vehicle;
//              int i, j, k;
//              int activeVehicles = 0;
//              int travelledVehicles = 0;
//              int wifiVehicles = 0;
//              long messagesCreated = 0;
//              long IDsChanged = 0;
//              double messageForwardFailed = 0;
//              double travelDistance = 0;
//              double travelTime = 0;
//              double speed = 0;
//              double knownVehicles = 0;
//              for(i = 0; i < regions.length; ++i){
//                      for(j = 0; j < regions[i].length; ++j){
//                              vehicles = regions[i][j].getVehicleArray(
//
//
//                              for(k = 0; k < vehicles.length; ++k){
//                                      vehicle = vehicles[k];
//                                      allVehicles.add(vehicle);
//
//                              }
//                      }
//              }
//
//              System.out.println("\n Total " + allVehicles.size() + " V
//        return allVehicles;
//
//
//      }

//      public void updateLog(){
//
//              try {
//
//                  writer.write("Something\n");
//                  writer.write("new line\n");
//              } catch (IOException ex) {
//                // report
//              } finally {
//                 try {writer.close();} catch (Exception ex) {/*ignore*/
//              }
//         System.out.println("File output over");
//
//      }

//      public void writeVehicleInfo(ArrayList<Vehicle> vehicles){
//
//
//              Vehicle vehicle;
//              int i, j, k;
//              int activeVehicles = 0;
//              int travelledVehicles = 0;
//              int wifiVehicles = 0;
//              long messagesCreated = 0;
//              long IDsChanged = 0;
//              double messageForwardFailed = 0;
//              double travelDistance = 0;
//              double travelTime = 0;
//              double speed = 0;
//              double knownVehicles = 0;
//
//              try {
//                      for(k = 0; k < vehicles.size(); ++k){
//
//                              vehicle = vehicles.get(k);
//                                  // steady id is fixed device id. wher
//                                      writer.write(vehicle.getStedyID()
//                                      writer.write(" ");
//
////                                    writer.write(vehicle.getID()+"");
////                                    writer.write(" ");
//                                      writer.write(vehicle.getTotalTrav
//                                      writer.write(" ");
//                                      writer.write(vehicle.getTotalTrav
//                                      writer.write(" ");
//                                      writer.write(vehicle.totalWaitTim
//                                      writer.write("\n");
//                              }
//
//              }catch (IOException e) {
//                      // TODO Auto-generated catch block
//                      e.printStackTrace();
//              }
```

```
//
//
//
//      }

//      public void logAllVehicleInfo(String fileName){
//              outputFileName = fileName;
//              logAllVehicleInfo();
//      }


//      public void logAllVehicleInfo(){
//
//              LogVehicleData logVehicleData = new LogVehicleData();
//              System.out.println("\n\n ****Going to call VehicleInfo\n"
//              // No parameter is needed. getting all the vehicles from
//              ArrayList<Vehicle> vehiclesList = logVehicleData.getAllVe
//              logVehicleData.writeVehicleInfo(vehiclesList);
//
//      }

        public static void initialFileWriter(String fileWriterFileName){
                try {
                        System.out.println("initiallizying␣file␣writere:␣
                        fileWriter = new FileWriter(fileWriterFileName, t
                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
        }

    public static void writeVehicleStreetInfo(Vehicle vehicle){
//      System.out.println("write vehicle street method is called");
        try {

//              System.out.println("in try method to write");
                        fileWriter.write("" + vehicle.getStedyID()+"␣");
                        fileWriter.write( "vtdis:" + vehicle.getTotalTrav
                        fileWriter.write( "vtt:" + vehicle.getTotalTravel
                        fileWriter.write( "vtwt:" + vehicle.totalWaitTime
////                    fileWriter.write( "vtspeed:" + (vehicle.getTotalT
//                      fileWriter.write("maxTF:" + vehicle.maxTrafficSig
//                      fileWriter.write("myTF:" + vehicle.myTrafficSigna
//                      fileWriter.write( "vsstops:" + vehicle.vehicleStr
//
//                      for(int i=0; i < vehicle.vehicleStreetInfoList.si
//                              VehicleStreetInfo vehicleStreetInfo = veh
////                            fileWriter.write( "sn:" + vehicleStreetIn
//                              fileWriter.write( "sl:" + (int)(vehicleSt
//                              fileWriter.write( "tt:" + (int)(vehicleSt
//                              fileWriter.write( "wt:" + (int)(vehicleSt
//
//                      }


                        fileWriter.write("\n");
                        fileWriter.flush();
//                      System.out.println("write vehicle street method h
//                      fileWriter.close();


                        System.out.println("log␣flushed␣to␣file");

                }
        catch (IOException e) {
                        // TODO Auto-generated catch block

                        e.printStackTrace();
                }

    }

        public static void closeFileWriter(){
                try {
                        fileWriter.flush();
                        fileWriter.close();
                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
        }


}
```

A.2 Vehicle route information Code

This class is used to find out details on how much time a vehicle is spending on a road. . .

```
        public void updateTrafficLightInfo(Street curStreet){
//              if(curStreet_.getStartNode().isHasTrafficSignal_()){
//                      TrafficLight trafficLight = curStreet_.getStartNo
////                    System.out.println("Start Node: This street has t
//                      curVehicleStreetInfo.maxTrafficSignalWaitTime =(i
//                      maxTrafficSignalWaitTime += trafficLight.getYello
//              }
////            if(curStreet_.getEndNode().isHasTrafficSignal_()){
////                    TrafficLight trafficLight = curStreet_.getStartNo
////                    System.out.println("End Node: This street has tra
////            }

        }

        public  void updateVehicleStreetLogInfo(Street curStreet_){
//              System.out.println("updateVehicleStreetLogInfo " + getSte
//                                              curStreet_.getName() + "
//                                      stopTime_);
                // negative means this is first street

//              System.out.println("\nroadsite unit size: " + knownRSUsLi
                if(curStreetEnterTime <= 0){

                } else{
                        curVehicleStreetInfo.totalTime = totalTravelTime_
                        curVehicleStreetInfo.totalWaitTime = totalWaitTim
                        if(curVehicleStreetInfo.totalWaitTime > curVehicl
                           maxTrafficSignalWaitTime     += curVehicleStre
                        } else{
                                myTrafficSignalWaitTime += curVehicleStre

                        }
                        vehicleStreetInfoList.add(curVehicleStreetInfo);


                }

                curVehicleStreetInfo = new VehicleStreetInfo();
                curVehicleStreetInfo.street = curStreet_;
                curStreetEnterTime = totalTravelTime_;
                curStreetEnterWaitTime = totalWaitTime;









        }

        // this method will be called when the vehicle reaches its destin
        public  void updateVehicleStreetLogInfo(){
                System.out.println(getStedyID() + "␣has␣reached␣its␣desti
                curVehicleStreetInfo.totalTime = totalTravelTime_ - curSt
                curVehicleStreetInfo.totalWaitTime = totalWaitTime - curS

                vehicleStreetInfoList.add(curVehicleStreetInfo);
                LogVehicleData.writeVehicleStreetInfo(this);
//              System.out.println("......written the street info to file



//              RSU[] rsus = null;
//              RSU rsu = null;
//              for(int i = 0; i < regions_.length; ++i){
```

```
//                              for(int j = 0; j < regions_[i].length; ++j){
//                                      rsus = regions_[i][j].getRSUs();         /
//                                      int size = rsus.length;
////                                    System.out.println("RSU amount: " + size)
//
//                              }
//                      }


                }

                // for thesis going to store vehicles every street info

}
```