Map Updates
a VANET application

Pavan S Gaonkar

Master of Science Thesis

Wireless and Mobile Communications Group (WMC)
Department of Telecommunications
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

TOMTOM®

Wireless and Mobile Communications

Map Updates
a VANET application

Master of Science Thesis

For the degree of Master of Science in
Wireless and Mobile Communications Group (WMC)
at Department of Telecommunications
at Delft University of Technology

Pavan S Gaonkar

May 30, 2012

Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Delft, The Netherlands

in association with TomTom International B.V

Delft University of Technology
Department of
Telecommunications

The undersigned hereby certify that they have read and recommend to the Faculty of Electrical Engineering, Mathematics and Computer Science for acceptance a thesis entitled

Map Updates
a VANET application

by

Pavan S Gaonkar

in partial fulfillment of the requirements for the degree of

Master of Science.

Dated: May 30, 2012

Supervisor:

Dr.RangaRao Venkatesha Prasad

Readers:

Prof.dr.ir.Ignas Niemegeers

ir.Jeroen Trum

Dr.Christian Doerr

Abstract

Digital maps, based on which navigation systems in vehicles operate, require frequent updates to reflect changes in real world. These changes currently being issued using off-line mechanisms are cumbersome in nature as well costly. Therefore a need exists, to come up with a wireless solution which can not only perform updates without human intervention but also be cost effective. IEEE 802.11p a standard, enabling vehicles to communicate with each other to form a Vehicular Area Network (VANET), is being pushed as a technology for future safety in vehicles. The high data rates and communication range capabilities of the technology lead us to investigate it as a viable solution for providing map updates.

Our thesis work is first of its kind where real world mobility traces are used in study of VANET. These traces, obtained from GPS devices used for navigation in vehicles, are collected for all vehicles moving in and around the country of Netherlands for entire 24 hours. No existing network simulators have the capabilities to work with them due to the enormity of the traces. In this article we propose our own simulator DC-PIC, which is simple and scalable in nature. It is designed using concepts of divide and conquer and particle in cell, which are widely used in the field of high performance computing.

The feasibility study of application of map updates involves investigation into the degree of support required from Road Side Unit (RSU). We analyze influence of various parameters of RSU on spreading updates to vehicles and based on which we propose values for these parameters suitable for our application. The empirical analysis of mobility traces is performed to provide insight into suitable locations for potential RSUs. Further, empirical analysis is conducted to obtain insight into mobility patterns and communication capabilities of vehicles in VANETs. Finally, we propose a simple and distributive dissemination algorithm named Adaptive and Distributive Broadcasting algorithm (ADB) which provides spread of updates comparable with that of flooding algorithm but without the unnecessary overhead of excessive redundant messages and collisions.

Table of Contents

List of Figures

List of Tables

Acknowledgements

It is a pleasure to thank many people who have made this thesis possible.

First and foremost i would like to thank Roland Van Venrooy from TomTom International B.V for providing me this opportunity to work in the field of VANETs. I am grateful to the company for all the assistance and especially for providing me the traces without which this thesis would not have been possible.

I would like to thank my daily supervisor ir.Jeroen Trum at the company for providing me guidance in technical aspects of the thesis, helping me to have a better understanding on the Map technology and the Simulator. His vast experience along with friendly nature has played a vital role in the completion for my thesis for which i am grateful to him. I also thank him for the efforts he undertook to review my thesis and provide suggestions for the same.

My gratitude to my daily supervisor Dr.RangaRao Venkatesha Prasad at the university is too big to fit in words. His enthusiasm, being easy to approach are few of the qualities because of which the duration of my thesis has been an excellent one. His sound theoretical and practical knowledge of communication systems, interesting ideas and pushing me to do excel more has made my this thesis a reality. I am also grateful for his assistance in reviewing and suggesting improvements for my thesis.

My special thanks to my close and dear friend Lokesh Gupta for being part of technical discussions, thesis writing and without whom the thesis would have been very difficult to complete. My regards to all dear friends at University and TomTom for bearing me during the difficult and challenging phases of the thesis and for providing moral support all the way.

Lastly and most importantly, I wish to thank my parents and grad parents for their unyielding love, support, teachings. To them i dedicate this thesis.

Delft
May 30, 2012

Pavan S Gaonkar

Chapter 1

Introduction

1-1 Vehicle to Vehicle Communication

Advances in communications have revolutionized the way we live our lives as an individual. Mobile phones, internet have become an integral part of our daily lives and one cannot imagine living without them. While at a society level, it has been a catalyst for growth and innovation for various sectors of life. One such sector which immensely benefited, is transportation with most innovations in the field accounted to advances in communications. The ability to digitally control entire vehicle, whether be the seating arrangement or engine control, all has been possible due to development of communication systems of Controller Area Network (CAN). Advances in mobile communications, have made it possible to extend the vehicle as a communicating device with the use of 2G/3G communication systems. Technology such as Bluetooth have enabled vehicles to be play music from MP3 players without any wires, GPS systems have enabled vehicles to travel without requiring prior knowledge of routes.

All the advances either in current use or proposed for the future share the same objective of providing driving safety and comfort for an individual. One such future technology is known as Vehicular Area Network (VANET). It is proposed to provide next generation safety in vehicles by enabling vehicles to communicate with each other. They are to be built with the capabilities of large range of communication, low setup time and large data rate so as to overcome the complexities associated with mobility. Further, they are proposed to provide capabilities to communicate with Road Side Unit (RSU), which will act as the necessary gateway to backbone network in order to provide services such as internet browsing.

VANETs are deliberated to have the potential to provide solutions to future vehicular safety, logistics and traffic solutions. These goals have brought a great deal of interest among research community and the industry. With the assistance from various private and government organizations, research has been conducted on various issues in

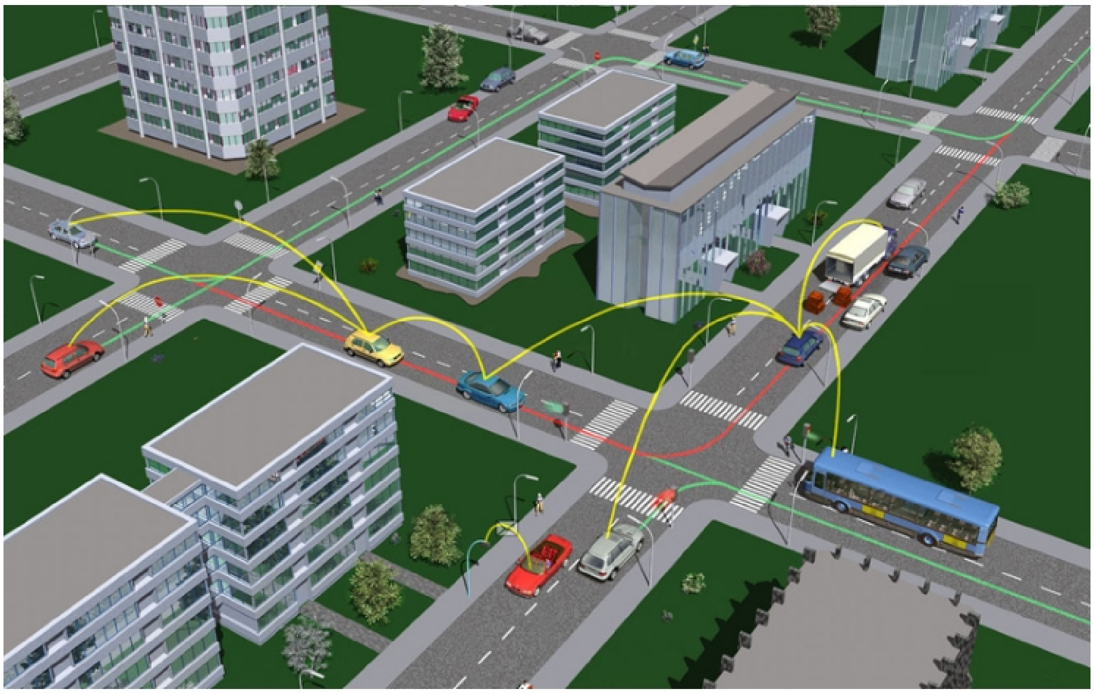


Figure 1-1: Vehicle - Vehicle Communication Systems [1]

VANETs, ranging from wireless communication characteristics, antennas, medium access layer and network layer protocols, simulation tools and so on. To provide a further boost and a unified step forward to achieving the above goals, IEEE has initiated a special workgroup to form a standard called IEEE 802.11p [2] focused on VANETs. One of the most important challenges for this workgroup was to design a medium access layer that can support a wide range of data rates. This has opened up many possibilities to design VANETs that are capable of supporting a wide range of applications and not just dealing with safety or traffic updates. To this end, various investigations on the feasibility and analysis of multiple types of applications have been conducted. A few prominent ones include parking information updates, advertisements, file transfers and even high bandwidth applications such as live video streaming and multiplayer online games on VANETs. Our work involves investigation of another consumer application of spreading digital map updates. In this thesis, we address specifically the communication between vehicle-vehicle and vehicle-infrastructure. This thesis takes the specific case of map updates. However, we believe that the analysis and the contributions of this work are not only limited to map updates and can be extended for applications such as Over the Air Software Upgrades. We study generic performance metrics (that would be expanded in Chapters 4) that are useful in various forms. In this chapter we provide the motivation for Map Updates as a application followed by objectives and end it with describing structure of our thesis.

1-2 Motivation

Digital maps form the heart of today's navigation systems. Direction advice to the destination, at early stages of the technology, was based on mere distance metric. These days it has moved into far more complex algorithms taking into account, parameters such as the static and dynamic nature of (real-time) traffic information, the weather and toll prices. This has enhanced the maps showing information about road networks, store the information on the type of roads, toll prices, locations of various services like fuel pumps, service station and in general any Point of Interest. POI do change often and more and more information is being added in the digital world. As more and more information is poured into the maps, they are more prone to become outdated. It been found that map changes occur at around 15% every year [3] [4]. These changes can be permanent such as construction of new roads and destruction of old roads or could be temporary such as closure of roads due to repair, or can have changes to services such as changes to POI. Any changes in real world if not reflected back in the digital maps will degrade driving experience. Hence to avoid this, all map service providers and automotive Original equipment manufacturer (OEM) periodically provide updates using various mechanisms to its customers for free or at nominal prices. Currently, deployed products have procedure of updating maps based on use of memory devices such as SD cards, CD/DVDs which are then plugged into the navigation devices, either being factory fitted systems or the ones brought off the shelf Portable Navigation Devices (PND) [5]. For some PND, one can plug-in the device into the internet where it can get the map update from central server [6]. This mechanism is not only cumbersome for the customer but also costly for customer and the map supplier.

A wireless mechanism to update with least possible human intervention is the best solution for resolving the difficulties associated with the process of update. One of the solutions would be, to make use of mobile communications such as 3G and 4G network. But the infrastructure costs, additional hardware installations, interoperability between various services providers and huge data requirements to issue map changes will result in high costs of map updates as an application, thus discouraging customers to embrace this mode of update. There is need to have wireless update service that is inexpensive for OEM or the map supplier which can be easily accepted by the customers.

As compared to mobile communication, use of IEEE 802.11p can not only help in communicating with infrastructure points but it can also support adhoc communication with other vehicles resulting in less dependence on the infrastructure points. This turns out to be an inexpensive solution for both the OEM supplier and potentially for the customers. Secondly, because IEEE 802.11p is being pushed as a mechanism to provide safety in next generation of vehicles, it would probably be a regulation to have all future vehicles fitted with this hardware. Hence, the map supplier would not require explicitly baring the entire expenses of setting up the hardware in a vehicle. These advantages make a valid proposition to investigate the map updates as an application based on IEEE 802.11p. However, our study is generic enough to be applicable to many other applications, which include software updates for infotainment unit inside vehicles, advertisements, news feeds etc.

1-3 Objectives

The objective of the thesis is to investigate the feasibility of map updates as an application for C2X Communication. As part of the study, we intend to investigate the following tasks.

T1: Feasibility of the use of real world mobility traces on existing simulator and discuss limitations with them. Develop a new simulator which can work with these traces and is scalable with any number of vehicles and trace data.

T2: The need of Road Side Unit (RSU) and other sources that act as the provider of updates to the vehicles.

T3: Requirements of the position and role of road side units have in achieving the required spread of updates.

T4: Infuence of number of units, time of broadcast and communication patterns has on the overall spread of updates.

T5: Propose a decentralized algorithm to achieve the possible spread of updates with least overhead and efficient mode of communication.

1-4 Structure of Thesis

This thesis is organized as follows: Chapter 2 consists of literature survey associated with various aspects of VANET consisting of applications, simulators and routing protocols. Chapter 3 discusses various requirements and aspects associated with application of map updates. Chapter 4 discusses the empirical analysis of mobility traces providing insights into the characteristics of mobility such as trend of active vehicle count, travel trip times. It also provides analysis of communication capabilities of mobility by finding the characteristics of connectivity, degree of connectivity of a vehicle in traffic. Chapter 5 discusses the limitations of existing simulators to work with traces of such high count of vehicles. To overcome these limitations we propose a new simulator. We put forth the design concepts, algorithms and implementation details about the simulator. Chapter 6 discusses the feasibility of vehicles or RSU as a primary source of map updates. It discusses influences of system parameters of RSU such as, number of units, location of these units and the time period of broadcasts have on the overall spread of updates. Chapter 7 discusses the design considerations of routing protocol used to spread the updates. It also compares the performance of the protocols with other protocols and finally conclusions and future work are discussed in Chapter 8 and Chapter 9 respectively.

Chapter 2

Literature Survey

VANETs being projected as the necessary tool to provide the next generation safety features in vehicles, has gathered lot of interest among the research community. Along with findings from various government and private institutes various aspects of VANETs have been researched. We in this chapter have a brief discussion on some of these researches related to our work and also showcase how our work offers differs from the work done so far.

2-1 Applications for VANETs

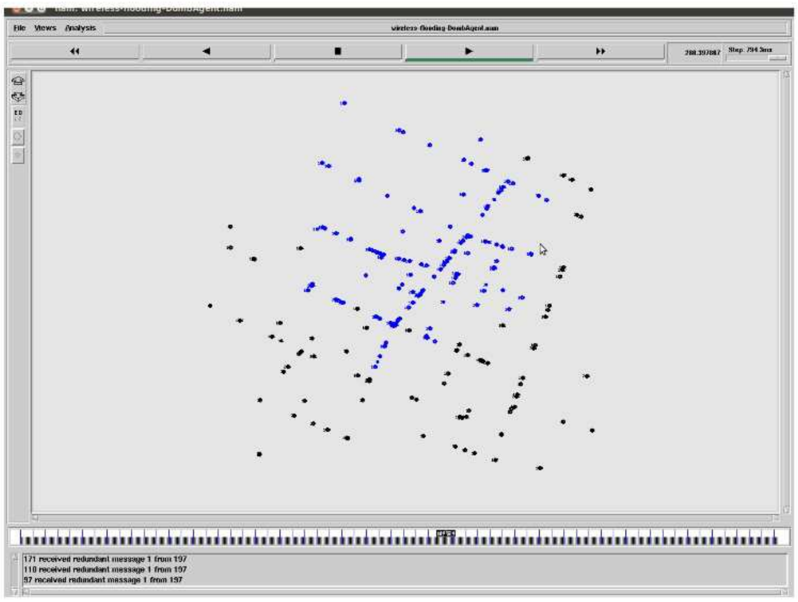
Fast adaptation of the technology among the public is only possible by providing a large set of multimedia or entertainment applications that can entice the customers to embrace and pay for this new technology. Understanding this importance the standards have proposed wide range of supported data rates so that the technologies can suite different applications. The importance of this set of applications is known across the research community and hence various different investigations have been undertaken across the globe. We list some of the major ones. A distributed broadcasting algorithm for finding free parking lots have been proposed in [7]. It proposes to use grid based system for data collection and distribution. It proposes to collect information about each locations based on data provided by parking automats and by the aggregated information provided by other vehicles. In order to save bandwidth requirements it proposes to make use of aggregation of information for farther distances so as to only provide approximation of information for far off locations. The paper puts forth decisions involving broadcasting rate for the Parking automat as well as for the inter vehicle communication based on analytical and simulation studies. Another application based on data aggregation is proposed in [8] which discusses the feasibility of search engine for VANETs with the use of flooding or distributed hash tables to distribute the information about locations of vehicles. VANET based multiplayer games is investigated in [9]. It discusses the issues unique to VANET with respect to multiplayer gaming and investigates suitability of traditional model of client server mechanism for multiplayer gaming in VANETs. The feasibility of MP3 streaming from a fixed source to vehicles over VANETs is studied in [10]. The use of VANETs for video streaming, with receivers being the vehicles while source could either be vehicles or any stationary sources is investigated in [11]. Use of vehicle to vehicle communication to collect information about patients driving around in vehicles using trusted vehicles such as police vans and paramedics for data gathering is evaluated in [12]. A content delivery and sharing mechanism for VANETs inspired from Torrrent like systems is in [13] and [13]. All these applications study the issues unique to the applications in VANET systems and propose solutions and provide simulation studies to prove the validity of their proposals. All the above applications discussed above do not consider a real world scenario as all of them rely on mathematical models to provide them with mobility patterns. They all are of the type where they assume to have a dense infrastructure support in form of RSU which is not possible during the early stages of deployment of the technology and hence fail to consider these scenarios for the analysis of their work. Above all, the applications considered here have analyzed their applications only for a small finite time (few minutes) over a small region (few kms) which is an issue because they fail to analyze their applications in real world scenarios where the traffic patterns, densities change over the entire day. We with the help of real world traces analyzing the performance of the application over the entire day overcome these limitations.

2-2 Simulators

Real world experiments for VANETs requires infrastructure such as Road Side Units (RSUs) as well as vehicles be equipped with the required hardware for communication. As there are very few test-beds with this required infrastructure as well not many commercial hardware solutions available for providing this communication systems makes real world testing not a viable solution. Hence most of the work done till date rely on simulation tools to analyze VANET applications and get insight into the research work. VANET simulation involves two aspects. (i) Network simulation and (ii) Mobility simulation. Network Simulation is the ability to simulate the communication protocol stack for IEEE 802.11p consisting of the physical constraints such as limited communication range, single access of radio channel. Mobility simulation deals with the ability to simulate movement of nodes under simulations which can depict real world movement.

2-2-1 Network Simulators

Network simulations have been actively used in study of various wired and wireless communication and so many of these tools itself have provided extensions to the standard tool package to provide support for IEEE 802.11p simulation. Over the time support of IEEE 802.11p technology by the tools have increased in number so selecting the right tool involves various technical and commercial aspects to be taken into consideration. The general technical aspects involved in decision making including the support for different routing protocols, propagation models, etc. The ease of adding new or modifying existing protocols, the development language used and the ability to generate detailed reports from simulation runs is also an important aspect. Besides these technical aspects the commercial aspects such as license types as well as costs involved also play a vital role in decision making.


nam: wireless-flooding-DumbAgent.nam
File
Views
Analysis
wireless-flooding-DumbAgent.nam
288.397887
Step: 794.3ms

```
171 received redundant message 1 from 197
110 received redundant message 1 from 197
97 received redundant message 1 from 197
```

Figure 2-1: View of ns2 for Flooding Algorithm on 1000 vehicles

ns2 [14] is a discrete event simulator used extensively in wireless communication research and in general to simulate communication protocols. The project initiated back in 1995 has been supported by various DARPA projects over the years. Being open source and free has made it easy for various teams around the world to provide contributions to the tools. Due to this it supports of one of largest set of TCP simulation, routing protocols for wired and wireless network, etc. Currently two versions ns2 and ns3 are in use with the latter being in the early development phase. The development of tools is done in C++ and OTcl. Open source and modular architecture has made modifications to the tool quite easy. The majority of IEEE 802.11p support has been provided by contributions from CMU Monarch Projects [15]. It provides an extension of NAM a front end tool providing graphical support for viewing simulation traces. Fig.2-2-1. shows a view of simulation of 1000 nodes in VANET for flooding algorithm. It provides support to various propagation models as well as to different mobility models.

Glomosim [16] is another tool which has gained some interest recently. It solves the issue of simulating large set of nodes. It uses a technique of aggregation with the help of which it can support a large set of nodes in our case vehicles for example. Tests executed show it supporting upto 100,000 vehicles on Super SPARC Machine with traces generated with mobility models. The shortcomings of the tool include that currently it supports very less set of protocols. Apart from that in order to change or add new modules basic knowledge of PARSEC Compiler is required and the biggest problem is it being only available free of cost to US Universities while the commercial version is being sold via [17]. Jist [18], stands for Java in Simulation Time, is another discrete event simulator which in recent years has gained a lot of interest due to ease of programming in Java and potential to work with extremely large set of nodes. It basically consists of a simulation kernel which provides simulations to be executed efficiently in time. It computes and provides transparent dynamic reconfiguration and inspection of simulation parameters. Along with SWANS [19] Scalable Wireless Adhoc Network Simulator which provides entire OSI stack modules that can be used to simulate various wireless communication protocols. Apart from these, other noteworthy tools to be mentioned are Qualnet [17], Opnet++ [20] and Omnet [21].

2-2-2 Mobility Models

Mobility being integral part of VANETs, makes it an important requirement for any simulator tool. Traffic simulators are used to generate mobility which is very important for analysis of simulation studies. Mobility requires consideration of road topologies, road characteristics, geographic influences along with considerations on vehicle to vehicle interactions and vehicle to road interactions for real world mobility simulation. Due to the concerns on the complexity involved in actual replication of real world mobility, accurate modeling requires larger set of parameters to be taken into consideration. This results in increased computation time as well as the need for higher processing power. This hinders the investigations as one has to rely on costly and sparsely available high performance computing machines. Hence to keep the computation time low various mobility models consider different levels of abstraction and try to provide simpler model and at the same time not wandering away from the actual mobility. Based on the granularity of parameters whether the parameters are attached to individual nodes (vehicles) or at a general traffic level, they are classified into microscopic and macroscopic models respectively.

Random Walk and Random Waypoint models [22] were the first set of models used for simulation of VANETs. As VANETs are considered a class of MANETS it was obvious choice to use the same models used in MANETS in VANETs. These models consider 2 parameters Speed and Direction for each of the vehicles. Based on the predefined bound random values of each of the parameters are calculated independent of other vehicles and then these vehicles are made to move. The obvious advantage it brings in is the simplicity in model ling it, but this in itself is also a disadvantage. As already stated vehicles do not move randomly but move along predefined road network makes this model inaccurate to simulate vehicle mobility. Apart from this as vehicles are driven by physical constraints one can have change of velocity without acceleration/deceleration adding to its inability to model vehicle mobility.

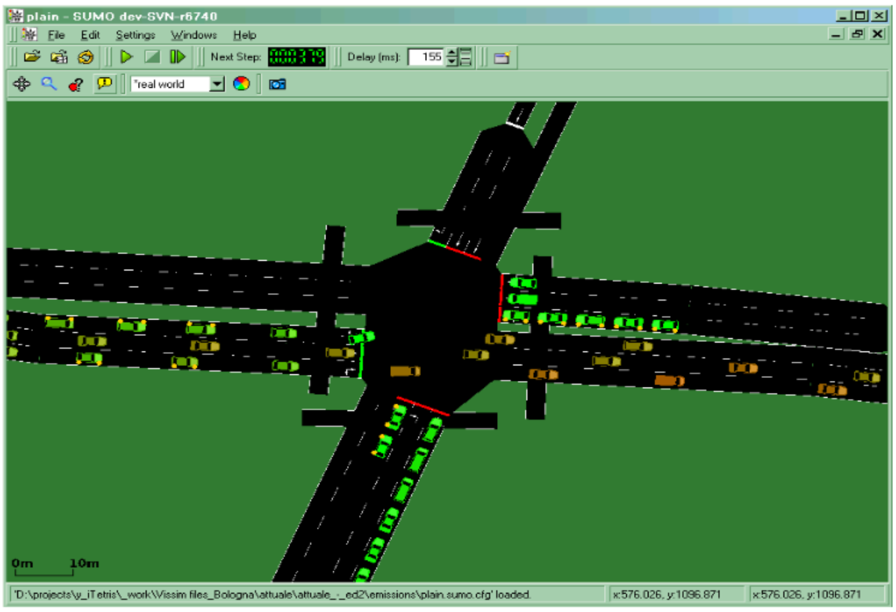
Manhattan mobility model [23] provides a solution to the problem of random movement. It considers a city scenario consisting of many road and intersections. It randomly places the vehicles in various locations along the road network and randomly decides the location and speed of travel. It restricts mobility along the path specified in the road network and uses Dijikstra‘s shortest path to route itself to end destination. Continuing on the advantages provided by Random Walk it tries to overcome the shortcomings from it. But this still does lack some key aspects which include the impact of traffic signals at junctions. All the vehicles in this model move ahead at crossing without needing to wait for crossing or slowing down which is not as seen in real world making it drift away from real world replication of mobility. Another aspect it fails to consider is the fact that vehicles will tend to slow down when multiple vehicles travelling along the same path adding to model being drifting away from the real world simulations. Similar to this a model to simulate multilane bidirectional highway was also proposed [24], which fares the same advantages and disadvantages as Manhattan mobility model. A new mobility model using Probabilistic Traffic Sign Model (PTSM) and the Traffic Light Model (TLM) was proposed in [25], which try to resolve the issue with regard to influence of traffic signals and road intersections in mobility. Solution to the other shortcoming was proposed in [26] where it proposes a new mobility model called as Reference Point Group Mobility (RPGM) where the group leader decides the effective mobility pattern. This model is useful in battlefield simulations and is also useful in VANET scenarios as and when vehicles come closer to from a group and the effective speed is determined by the speed of the leading vehicle.

2-2-3 Traffic Simulator

In order to suffice the ever increasing complexities involved in mobility models a dedicated software tool called Traffic Simulator was proposed. They provide the flexibility to choose right mobility model from scenarios to generate the traffic. This tool also provides features to make use of real world maps, either propitiatory or open source ones such as Tiger Maps or Open Street Maps to provide more realistic simulation. Because it is to be used along with network simulators the outputs from these simulators are form standard movement traces as required by the various renounced network simulators. Some of the prominent ones being Trans [27] with SUMO [28], Vanet MobiSim [29], Vissim [30] and STRAW - STreet RAndom Waypoint [31]. The Fig. 2-2-3. shows a GUI for SUMO for multi lane road junction.

2-3 VANET Routing Protocols

VANETs being characterized by fast moving nodes, disconnected network and vast region of interest provide unique set of challenges for routing protocols. Over the years various different protocols have been proposed, a few being adaptation of protocols studied in the field of MANETS [32], some adapted from the generic ad hoc routing protocols. Over the years various improvements have been proposed taking advantage of features unique to VANETs such as the hybrid structure of VANETs, inherent packet movement due to mobility of vehicles, predictive mobility of certain traffic and so on. Study of various parameters such as throughput, latency, communication overhead and



```
'D:\projects\y_iTetris\_work\Vissim files_Bologna\attuale\attuale_-_ed2\emissions\plain.sumo.cfg' loaded.
x:576.026, y:1096.871
x:576.026, y:1096.871
```

Figure 2-2: Graphical Interface for SUMO [28]

scalability are a few of the parameters that are studied and compared by these routing protocols.

Mobility has been an integral part of VANETs, plays a significant role in performance analysis of routing and other algorithms. Studies have been conducted which utilize the mobility to their advantage in making routing decisions. Fig. 2-3 shows improvements when routing decision can utilize vehicles moving in opposite direction to update vehicles, studies conducted in [34] show the use of predictive mobility obtained from public transportation systems to route packets to the required destinations. Road junctions which provide an excellent opportunity to meet vehicles traveling to and fro from varied directions makes them to be of importance for routing decisions. An example is [35], [36], [37] where they propose an routing algorithm in which vehicles at the junctions play a different role in routing as compared to when it is plying on other parts of roads. They utilize GPS coordinates to find whether or not vehicles lie at the junctions. Navigation systems of vehicles are utilized in [38], [39] to find the best suitable vehicles to deliver a packet to the required destination.

Delay Tolerant Networks (DTN) a class of networks, where the nodes carry forward the packet until it meets a suitable candidate to forward to. This class of network utilizes the fact that as vehicles move around the packets will travel along with them and hence make a decision to delay the forwarding of the packets until the required destination has been reached or have found other vehicles which could fulfill the requirement in a better way. This class of network is found to be quite useful in applications where time constraint is relaxed. GeoCast protocol utilizes geographic factors to make routing decisions. In this, they deploy techniques such as the farthest vehicle from the broad-

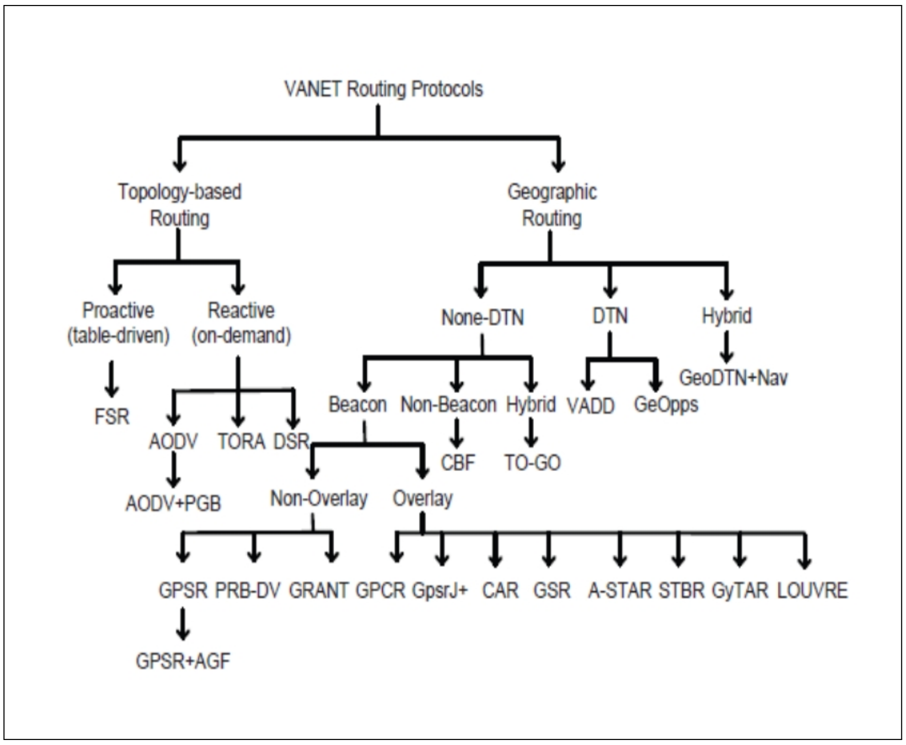


Figure 2-3: General Classifications of Routing Protocols [33]

caster to be the desired destination. Use of geographic knowledge helps to reach the destination in the quickest possible time when the traffic is high but fails in situations where the traffic is extremely low. When traffic is low use of this technique can result in scenarios where the same vehicle is provided the opportunity to be the broadcaster who was it some hops before. This results in a looping back which in turn hampers the performance of the algorithm. This issue is also known as looping error. To resolve it some modifications [37] have been proposed over the standard geo-casting protocols. Another issue they face is that considering only distance is not beneficial because more than the distance the traffic between the source and destination is crucial to get the fastest possible route to the destination. Hence few studies have modified to make use of traffic information along with the distance to obtain the required packet delivery.

2-4 Data Dissemination

Data dissemination technique is generally deployed when all or most of vehicles in the network require the information. Some of the examples include spread of advertisements, accident or hazardous event notifications. They are also used in various protocols for network maintenance such as neighbor and route discovery. All of these routing algorithms basically work on the principle of the source broadcasting the data, when the intermediate vehicles receive the data whether they are interested parties or not. But they will broadcast the data further until it reaches the intended vehicles.

The simplicity of broadcasting is the advantage which does not require large overhead of setting up and maintaining communication links. This advantage comes at the cost of it being inefficient and not being scalable in nature. Data dissemination achieved with simple broadcast mechanism is considered to be not scalable and hence is usually avoided to be used in its simple form. The reason for this is that, broadcasting is done by all the nodes until the broadcast remains active, as time progresses more and more vehicles become updated with the data and in turn start acting as a broadcaster. This results in an exponential growth of broadcasters along with exponential increase in number of broadcasts. This also increases the chances of collisions of broadcast of packets. The problem is also known as broadcast blast. So to resolve these issues various strategies have been deployed primarily focusing on who would be the next best suited node for broadcasting the data. Techniques which involve exchange of information with its neighbors include strategies like exchanging the list of neighbors. Thus the current broadcaster can decide the next broadcaster based either one with the most unique or the one with the least common list reducing the chances of collision. Another technique studied is, that the vehicles in the farthest direction take the responsibility of broadcasting the update. So in order to do this vehicles exchange their locations with neighboring vehicles. Thus the broadcasting vehicle receives the information from all the vehicles and then it broadcasts the update with the field informing who the next source of update would be. This technique, even tough quite useful, can be problematic in situations where traffic is heavy leading to exchange of lot of information amongst neighbors. Thus it takes a long time and as such by the time the current broadcaster decides on next broadcaster the current list of neighbors could become invalid. Another technique proposed [40] is to maintain a tree of distributed VANETs with vehicles at the extreme locations only allowed to broadcast the update making the issues of collisions to be kept at a minimum. Due to mobility characteristics the vehicles are very likely to leave and join the network and hence it proposes solutions to merge and split the network as and when required. Techniques based on probability, time slots and received signal strengths is proposed in [41] to decide on future broadcasters.

Another source of problem with multiple broadcasts is the issue of redundancy. Because of no exchange of information and due to restricted mobility along the roads, chances of same information receiving from the same source or from different source is much higher. Hence in order to resolve this there are multiple techniques proposed. Direction of travel to differentiate between the kind of messages broadcasted and the potential sources for the broadcast is used in [42] to make the broadcast effective. Use of Bloom Filters [43] and CRL [44] are deployed when there more than one item to update or multiple versions of the same item are of interest.

2-5 Our Contributions

Real world traces of mobility provides us unique opportunity to test VANET applications in scenarios if real world mobility without actually having the vehicles to travel with hardware with communication. This is unique for our application as we are able to analyze the application on real world movement and not rely on the mathematical models which have been used in analysis of other applications. We also later propose a new simulator which is capable to handle any set of traces, which is a limitation on the existing simulators. Study of parameters of RSUs in our application, where we analyze the effects parameters such as the suitable locations, number of units, the start time for broadcast and duration of broadcast have on the performance of the application is first of its kind. Finally the dissemination algorithms used to spread updates in VANETs are few and rely on extremely simple techniques such counter based or p probabilistic algorithms or use complex systems such as maintaining a tree. We propose a new technique in which, rather than considering a global perspective such as counter based or ‘p‘ probabilistic based algorithms and at the same time avoiding the overheads of maintaining the tree, we use local information to modify individual parameters of broadcasts so as to suit its local network topology. This algorithm is then tested on the same real world traces and the developed simulator where we show the improvements obtained by the use of the algorithm over the other standard algorithms. No research before this has been conducted where an algorithm which has been tested for such variations on traffic and for entire 24 hours, therefore makes our work unique as compared to research conducted so far.

2-6 Conclusion

We showcased some the work done so far by fellow researchers in the field of VANETs and tried to show how our work differs from all of them. Following chapters would describe the application of Map Updates in more detail and followed by description of our work done in it.

Chapter 3

Map Updates - Application requirements

Chapter 1 discussed the motivation of having map updates as an application for VANETs explaining the shortcomings of the existing implementation of issuing map updates and how VANETs can resolve the problem. In this chapter we discuss features of Navigation Data Standard (NDS) map format critical for our application. We discuss two types of updates that are supported by NDS and compare both of them. Further, we discuss the trends with respect to data sizes that an application would impose on the mechanism of spreading updates. Later in the chapter, we discuss different types of map changes that could occur, imposing different set of requirements of the spread of updates.

3-1 Navigation Data Standard (NDS)

Navigation Data Standard (NDS), provides a standardized runtime format for map data for map-based, in-vehicle applications which include Navigation systems, Stand-alone & Connected (Cooperative) Advanced Driver Assistance Systems systems and In-vehicle Travel Guides & eCommerce applications. It is an initiative started by German car manufacturers in 2005 with support from most of the leading map providers. The key feature of NDS is storing of the maps in the form of tiles and allowing changes to tiles without requiring changes to the rest of the map. An example is shown in Fig. 3-1. This feature enables updates to be issued in the form of providing new tiles for the regions where the map has changed. This mechanism is an immediate advantage over proprietary formats where any small change in a map would require updating the entire map. Apart from this standard process of NDS also supports incremental updates where updates are issued in the form of differences between the new tiles and the previous version of tiles. Incremental updates provide the advantage of only issuing the differences making data size small and thus data rates required for communication would be far more relaxed in nature. This advantage comes at the cost of complexity associated with the use of such mechanism. Scenarios where all the vehicles have same version of the map, any changes issued with the help of incremental updates would work without any problems. But in cases where vehicles have different version of maps incremental update would fail as the vehicle would not understand the changes as it would have a different version of current map from the one which broadcasted the update. To resolve this, the receiving vehicles need to update broadcasting vehicle with the version of the map it currently has so that the broadcaster can use this information to decide on which data to be broadcasted to the receiver. Situation gets worse when there are multiple receivers with different versions of maps as compared to that of broadcasting device where such a situation would enforce an individual reply to all the receivers resulting in degradation of performance of updates.

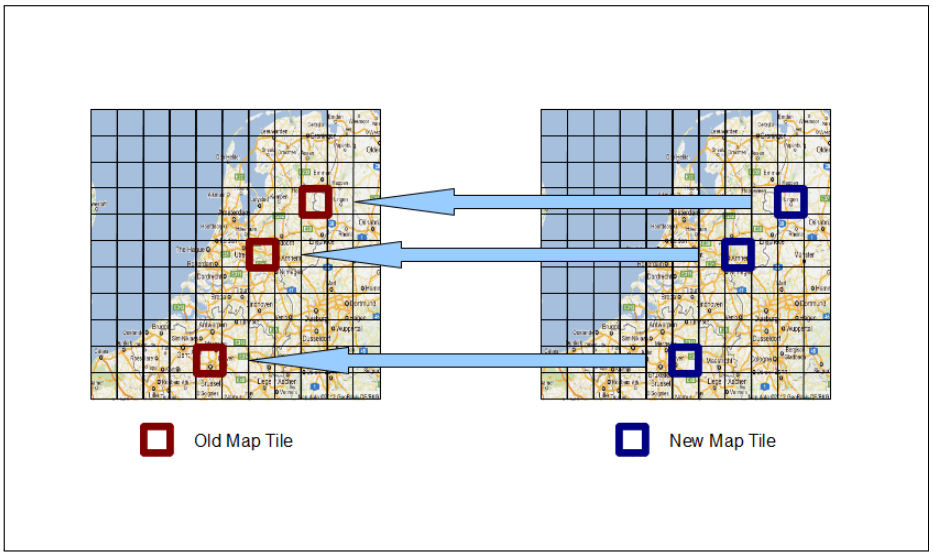


Figure 3-1: Updates in NDS [45]

NDS uses WGS 84, a World Geodetic System, with which it is possible to identify any point on Earth's surface uniquely by x, y co-ordinates with x corresponds to longitude and y corresponds to latitude. The origin is at the intersection of meridian and the equator. It defines these coordinates as 32 bit signed integers with a smallest unit defined as 90 / 2 raised to 30. NDS divides the whole map of the earth into tiles represented in hierarchical order, with a union of tiles at each level being capable to cover the whole surface of the earth. In all there are 16 levels with level 0 as the highest and level 15 as the lowest. Level 0 has 2 tiles with Tile 0 as the region represented using 0 to 180 degrees in longitude while 1 representing 0 to -180 degrees. Level 13 has been suggested to be used for map display. It uses these 2 co-ordinates to generate a Morton Code, a 64 bit positive integer obtained by interleaving x and y co-ordinate bits to represent the tile which is also used as Tile identifier. Table 3-1 showcases the sizes of tiles for different levels.

Table 3-1: Size of Tiles in NDS Map Format

Level	Length#kms	Breadth#kms
Level 0	20004	20037.58
Level 5	625.12	626.17
Level 10	19.53	19.56
Level 14	1.22	1.22

3-2 Application Requirements

3-2-1 Data Sizes

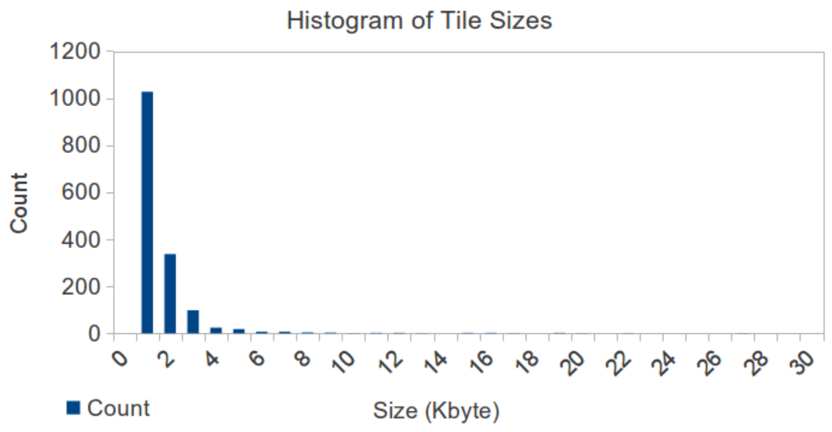


Figure 3-2: Sizes of tiles in NDS

One of foremost and crucial parameter associated with map updates is the amount of data that would be required to be exchanged for issuing update. As the road network is not consistent over the entire region, the map sizes will lack consistency. Similarly, the tiles in NDS too will not have constant data sizes. Hence, for issuing updates with NDS to an individual tile, it is vital to find data sizes of the tiles. Fig. 3-2-1 shows analysis of this for the region of Netherlands. The most of the tiles are in order of 10Kbytes with the maximum size of a tile being 270Kb.

3-2-2 Relevance of Update with Area

The map changes, even though are associated with a particular location, the importance of the update is over larger distances as compared to those with scenarios associated with traffic dissemination or traffic alerts. The reason for such larger distances is that, a vehicle even though far off would require this information in future.

3-2-3 Relevance of Update with Time

The map updates as an application impose, resulting in the relevance of data generated last for longer duration, unlike applications such as traffic dissemination or traffic alerts where the relevance of generated information is of few minutes. The primary reason for this is, the changes issued are of the permanent nature. Therefore, one requires the update to be issued to all the vehicles which would pass through it at any time of year. Even in situations where the changes issued are of temporary type, the changes are of relevance for few days.

3-2-4 Support of Road Side Unit (RSU)

Application of map update requires Road Side Units to provide the changes to the VANET as the map updates are complied outside the network. At the same time we intend to utilize a minimum set of Road Side Unit (RSU)s to keep the support from these units to a minimum level. Thus, it is crucial to place Road Side Unit (RSU)s at best possible locations. However, this cannot be calculated using the standard mobility models as they do not replicate the real world mobility associated with a specific region but a general model of mobility.

3-2-5 Reactive/Proactive Protocols

Application of map updates would always have the number of vehicles to outnumber the number of road side units. Therefore,, use of reactive routing protocols would imply inefficient mechanism where large number of requests originating from vehicles is directed towards the Road Side Unit (RSU)s. Apart from this, as the changes to the maps occurring at the infrequent intervals makes it very likely that a request for checking an existence of the update ends up with a negative response. On the other hand, making use of proactive routing protocols, where the broadcaster itself makes the decision to issue the update or not, is beneficial as in this case it is the RSU and the vehicles which have received the update from the RSU. Use of these protocols helps to avoid the overhead of unnecessary inquiry of updates as map change occurances are infrequent in nature and none of the vehicles nor RSU would have the knowledge of having a map updates until they have been issued by the central map server.

3-2-6 Type of Updates

The map changes can be categorized into multiple types of updates such as, permanent, temporary, changes that deal with road network changes or with data associated location based services such as changes to POI data or even based on the type of road changed. To adhere to different map updates the spreading algorithm must be configurable so as to be to able to handle different updating requirements.

Table 3-2: Types of updates

Parameter	**Tile Update**	**Incremental Update**
Size of Updates	Constant for the same Tile	Different even for the same tile
Additional Information	Only TileId required	TileId and version Id
Communication Procedure	Simple	Complex

3-3 Mobility Traces

The PNDs from TomTom have the capabilities to gather information about the movement of vehicles by keeping track of location and time. When these devices are connected to the internet, the data is collected back by central server to gather more information on road statistics such as speed profiles for the roads. This information is used as an input to enhance the services to the customer. In order to maintain the privacy of the user, all the statistics that can lead a particular trace back to the user are filtered.

3-4 Conclusion

In this chapter, the author has explained concepts related with application of map updates and the associated requirements. These requirements form a basis for our work to be followed which includes simulator implementation, empirical analysis of traces, VANETs system configuration and finally the data dissemination protocol.

Chapter 4

Empirical Analysis

Mobility plays a key role in performance of VANET systems,therefore a good understanding of it is required for better design of the system and the dissemination algorithm. This chapter analyzes vehicular mobility by performing empirical analysis of mobility traces to obtain details such as traffic count, churn, trip times and etc. It also includes, analysis of communication capabilities of a vehicle, which is studied to investigate the influences of sizes of map changes has on a vehicle‘s connectivity. The empirical analysis is conducted for vehicular traffic of entire country of Netherlands for 24 hours starting on first August 2011 at 2.00 AM. To the best of our knowledge, analysis of such an enormous data has never been done before and hence the data presented here is the first of its kind providing an understanding of mobility aspects of the traffic.

4-1 Traces

Traces used in this research have only come from vehicles using TomTom devices and not the entire set of vehicles that are running in Netherlands.This is crucial for the empirical analysis, as with the increase in the number of vehicles the associated statistics will also increase. The statistics would change only in magnitude and not in their general characteristics and hence the conclusions generated from preseted results would remain the same even if we used the entire set of vehicles running in and around Netherlands. Further, the service would be applicable only to those vehicles which are fitted with PNDs from TomTom. No other vehicles could register for updates from TomTom and nor would participate in dissemination of the map updates. Hence, we believe the other vehicles would impose no such problem or provide any assistance to our application

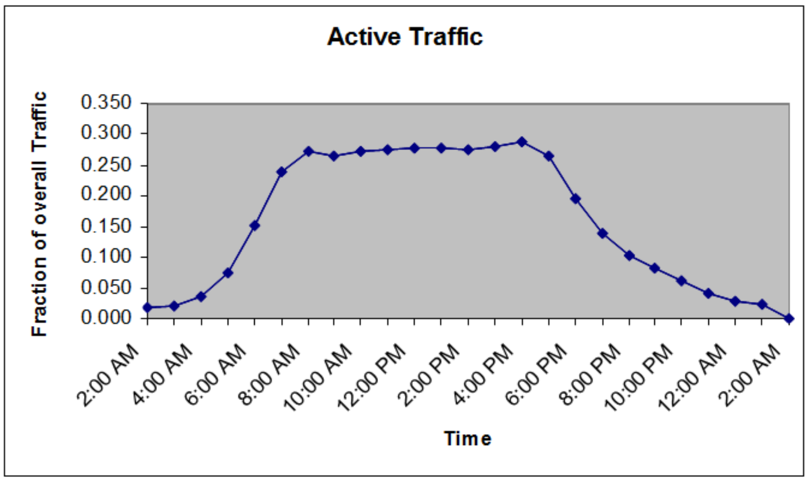


Figure 4-1: Active Traffic

4-2 Active Vehicle Count $C_{active(t)}$

The term Active Vehicle Count $C_{active(t)}$ stands for vehicles which are currently running around the country. The empirical analysis of this parameter shown in Figure 4-2 reveals that the parameter varies over time with low counts at early and late hours and highs during the day time. Another interesting fact is the peaks observed around the start and end of work time implying that maximum number of vehicular traffic is utilized to traveled for work. Figure 4-2 also reveals that maximum traffic reached that is around 27,000 vehicles which is nearly one fourth of the overall count of vehicles observed over the 24 hours. This parameter reveals the extreme variations observed in active traffic. Further, it reveals the sharp rate of change of traffic observed around the start and end of work. This evidence suggests that the dissemination algorithm should be capable of handling low and heavy traffic, as well as adapt quickly to rapid traffic changes at the start and at the end of work.

4-3 Rate of New Arrivals of Vehicles

New activations add to the existing network load and hence it is crucial to understand the rate of new activations occurring throughout the day. In the previous section we only gave an indication as to what is the current operating number of vehicles. Thus, it is insufficient to handle the need of new vehicles that are required to be updated. Here we provide an insight into the aspect of new arrivals of vehicles such that these vehicles induce more update requirements. We conducted an analysis of data to find the trends of new activations. Figure 4-3 shows the result of the trend of new vehicles getting activated over the entire day. The figure also reveals the trend observed with a S curve

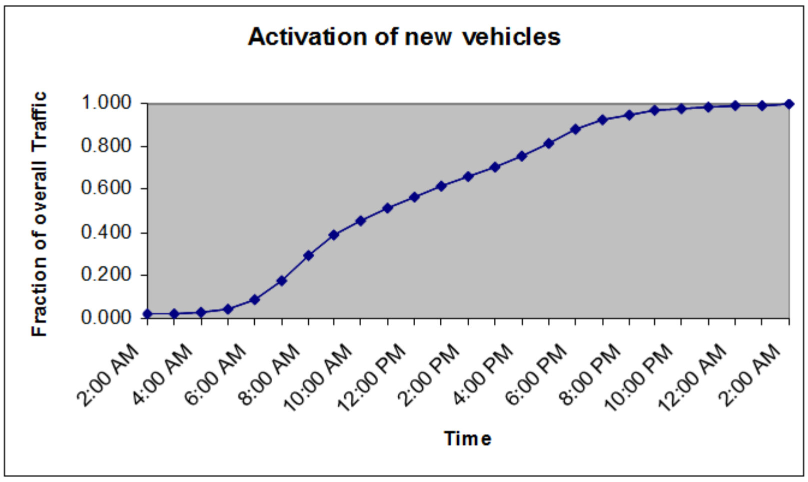


Figure 4-2: Arrivals of New Vehicles

having early and late hours being the ones with least number of new vehicles being activated. Morning hours reveal the largest additions of new vehicles with further rate additions slowing down as the day progresses. It also suggest that algorithm and the system itself is able to provide updates over the entire 24 hours if one wishes to spread it to maximum set of vehicles.

4-4 Churn in VANET α_t

Vehicles are used by human beings for different purposes and hence we end up having travel patterns with different travel times and travel destinations. Vehicles participate in VANET only when they are traveling, start participating in VANET when they start their trip and end participation when they reach their destination and turn off the vehicle. Different set of start and end trip timings present continuous change to the set of vehicles that are active over any time interval. These changes resemble the dynamics of peer participation in peer-to-peer (P2P) systems. We term this parameter as churn for VANETs. As with P2P systems, *churn* plays a crucial role in design of a system so does, in our VANET system for map updates and hence the study of this parameter This churn consists of two aspects: rate of addition and rate of removal of vehicles from the network. Each of these parameters is calculated as fraction of active vehicles and the result is shown in Figure 4-4. The figure reveals three different characteristics namely, in early mornings the rate of joining the network is superseding the rate of leaving the network, day time the rates remain fairly constant while at the end of day rate of leaving the network is superseding the rate of joining the network.

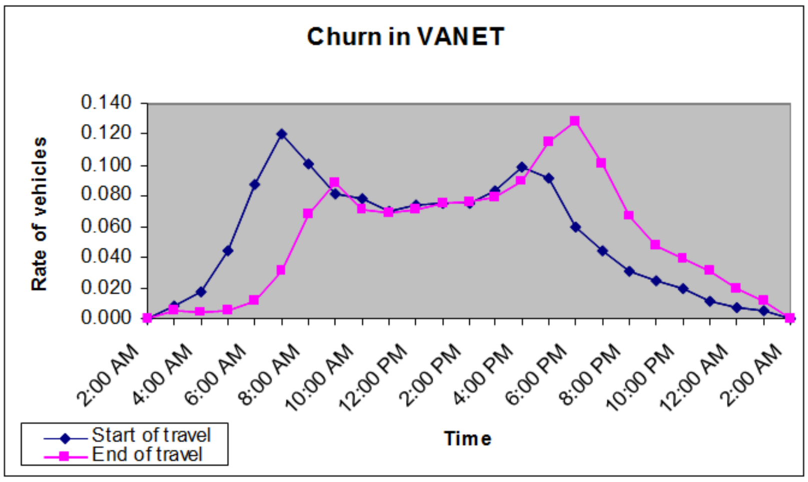


Figure 4-3: Churn in VANET

4-5 Departures from VANET

Vehicles will participate in VANET during the travel to their destination. The traffic data used is of 24 hours. We would have vehicles which will travel for more than one trip. Such vehicles which could not receive the update during the first trip have chance of getting updated in their next trip, while the vehicles with the update can boost the system capabilities when they start their next trip and act as another broadcasting node. We cannot foresee whether a vehicle will restart in later part of the day hence we use the statistics to present a mathematical model or lookup table which can be used to determine it. This will help us determine the routing parameters of individual vehicles based on its trip end time. The parameters are statistically found using the three graphs each for general departure from VANET, permanent and temporary departure from VANET. The general departure finds the number of vehicles finishing their trip during the given time interval, permanent departure estimates how many of the general departures will be of the type which never restart for the remaining part of the day and finally temporary departure infers how many of the general departures will be of the type which will restart again in remaining part of the day. The Figure 4-5 reveals that until the morning hours of the day the fraction of vehicles that would restart again has less than 50% chance and as the day progresses the probability of restarting again diminishes.

4-6 Travel Times

Map update, as an application for VANETs behaves as a P2P system responsible for updating vehicles with the map changes. Vehicles can participate in VANET communication only when it travels, so apart from calculating start and end times of travels, the

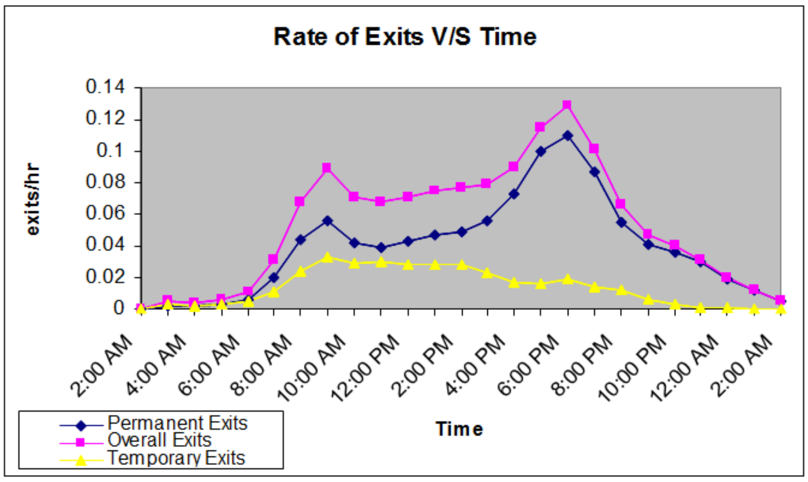


Figure 4-4: Exits in VANET

duration of travel is also crucial factor that will determine the spread of updates. We calculate the travel times for the entire traffic and showcase its histogram in Figure 4-6. This provides an indication on the distribution of active periods of vehicles. The figure reveals that the maximum traffic to have its trips of around 30 minutes with nearly 80% of entire traffic having individual trips with travel duration of less than 3 hours. We can conclude that we need information spreading algorithm which can ensure updates as fast as possible with average updating time around 30 minutes so that the spread of information reaches the largest fraction of traffic.

4-7 Disconnected Vehicles

Time	Fraction of Traffic
1	0.075
2	0.132
3	0.147
4	0.163
5	0.181
6	0.198
7	0.218
8	0.241
9	0.267
10	0.298

Table 4-1: Disconnectivity in VANET w.r.t Link Availability Time

The vast areas of road network to travel through, varied different time instances over

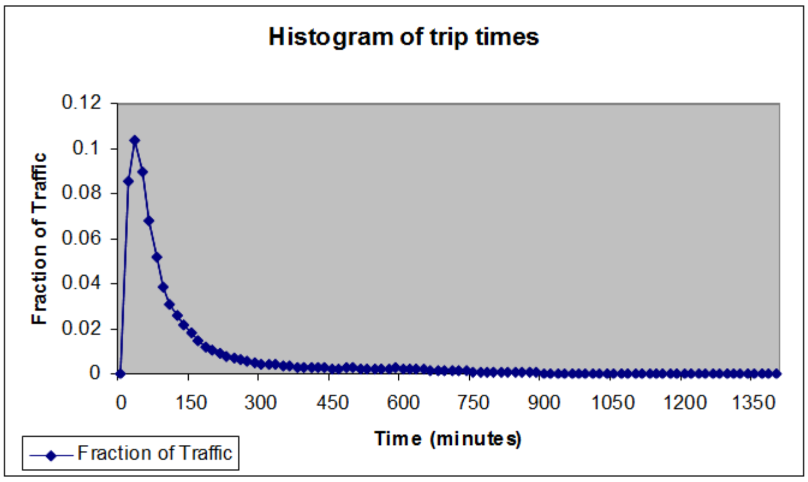


Figure 4-5: Travel Times for Vehicles

which vehicles can start and end their trip makes it very much possible that vehicles have their trips such that they never are able to communicate with any other vehicle throughout its trip. The importance of studying disconnected vehicles is that, it provides the upper limit of traffic that can be updated using V2V communications. This limit provides a real world benchmark to which the performance of dissemination protocols can be compared. This parameter also assists in making decision whether one can perform map updates in VANETs primarily based on V2V communications with least support from RSUs. The ability of a vehicle to communicate with other vehicle will be based on the fact whether two vehicles are within the communication range for the entire time required to communicate the complete map changes (in general, any update, or file exchange, etc.). Therefore we analyze the traces to provide the fraction of traffic that will never be updated for different time settings of 1,5 and 10 seconds. The purpose of using different time intervals is to simulate different data sizes required for transfer between the vehicles. The results are generated with fixed communication range of 350m are shown in Table 4-1 and its graphical representation depicted in Figure 4-7. Analyzing the figure, we can conclude that as time required for exchanging data increases, the fraction of traffic that will be disconnected too increases with nearly 30% of traffic unable to communicate with any other vehicle when data exchange requires 10 s are required for data exchange, while the same count reduces to 7% when the time required is only 1 s.

4-8 Connectivity of a Vehicle

We define the term, Connectivity of a Vehicle as the ability of vehicle to meet other vehicles and measured as the number of vehicles it has the potential to communicate. This term provides the capabilities of a vehicle of coming in proximity of set of vehicle

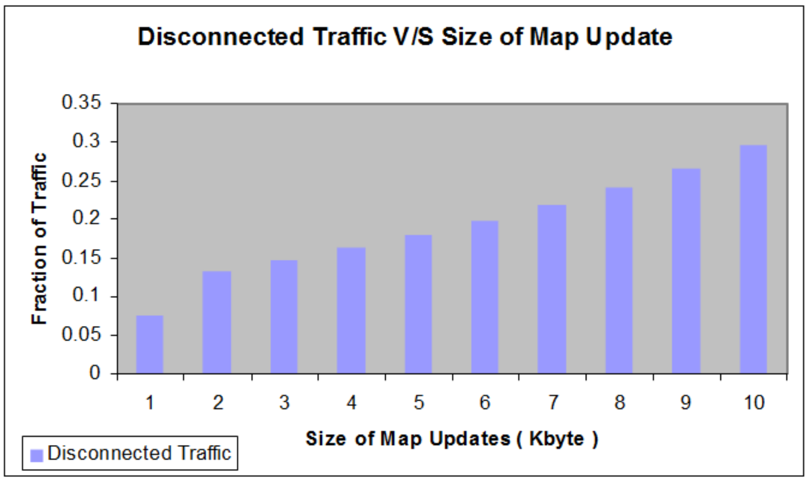


Figure 4-6: Disconnectivity in VANET

to which it is able to communicate the map changes. This term if found high amongst all vehicles, would indicate that every vehicle is highly likely to be connected with the rest of the vehicles and hence one could achieve the required spread with any one from the entire set of vehicles. Whereas a low value would imply that the VANET is highly disconnected one and requires large count of vehicles to meet the required spread of updates.

Link availability time is the time for which vehicles remain within communication range so as to update vehicles with the map changes. As map changes vary in sizes, so would the required link availability time. Analysis is conducted to calculate the number of instances of simultaneous communication for different settings of Link availability time (1/5/10 seconds) with communication range as 350 meters. The results are shown in Table 4-2 while a graphical representation is shown in Figure ?? The figure reveals

Time	**>=1**	**>=2**	**>=3**	**>=4**	**>=5**
1 sec	0.925	0.872	0.832	0.799	0.771
5 sec	0.819	0.732	0.662	0.603	0.551
10 sec	0.703	0.543	0.431	0.349	0.289

Table 4-2: Connectivity w.r.t Link Availability Time

that connectivity amongst traffic is better when the Link Availability Time requirement is less. The figure also reveals that ability of traffic to communicate to more number of vehicles decreases rapidly, in an exponentially decaying manner for larger time requirements (10 s) while as the time requirement decreases, the ability of traffic to communicate with other vehicles decrease more linearly in nature with diminishing slope. As part of conclusion, one can say that as the requirement of link availability time increases the ability to communicate and thus the ability to spread updates

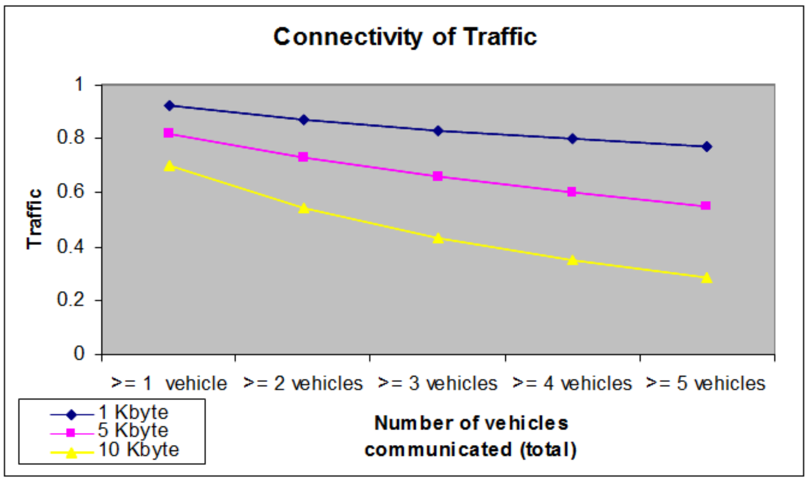


Figure 4-7: Histogram of Connectivity of Traffic

diminishes.

4-9 Degree of Connectivity of a Vehicle

We define Degree of Connectivity as the ability to communicate updates to multiple vehicles at a given instant. This parameter is crucial for our application as it determines the rate of spread of updates a system can accomplish. With map updates as an application where once a vehicle receives update it starts acting as a source of update. The ability of the system to update depends on the number of sources available at any instant. A higher degree of connectivity ensures more number of vehicles are updated at a time which ensures more number of sources are created and eventually achieving a faster rate of update. Connectivity in VANET is influenced by the duration for which the link is available for communication and so we analyze Degree of Connectivity for different configurations of link availability time. . Three different settings of time

Time	>=**1**	>=**2**	>=**3**	>=**4**	>=**5**
1 sec	0.866	0.813	0.767	0.725	0.687
5 sec	0.758	0.617	0.503	0.417	0.350
10 sec	0.545	0.328	0.211	0.143	0.102

Table 4-3: Degree of Connectivity w.r.t Link Availability Time

duration (1/5/10 seconds) of link existence has been considered to observe its influence on the maximum degree of connectivity of a vehicle The results are shown in the form of an histogram in table 4-3 with graphical representation in Figure 4-3. The figure reveals that the fraction of traffic capable of having a particular degree of connectivity

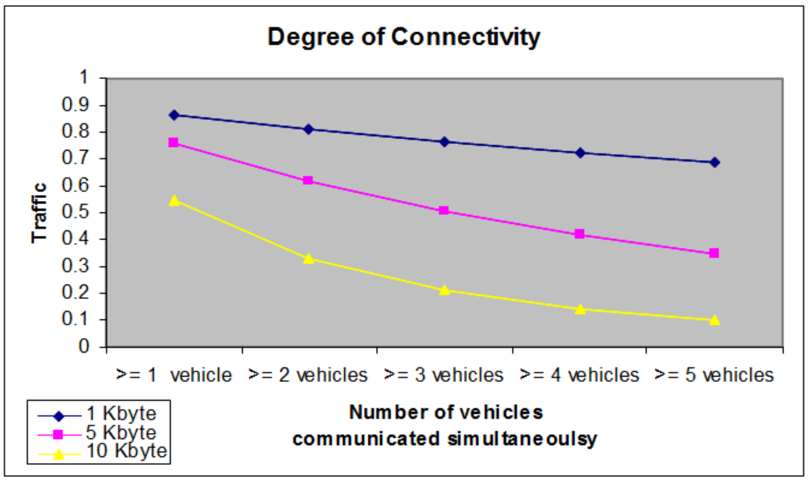


Figure 4-8: Degree of Connectivity

reduces rapidly, in an exponential manner for configurations which requires 10 s of link existence between vehicles for map update whereas the configuration which requires only 1 s shows only a linear decrease with minimal slope. This concludes that larger sizes of map changes would negatively impact the capabilities of VANET to disseminate the information of map changes.

4-10 Number of Instances of Simultaneous Connections

In previous section we studied the degree of connectivity parameter of a Vehicle indicating the capability of updating number of vehicles simultaneously. The study revealed more about mobility but fails to provide information on how many times the same vehicle was possible to communicate with the same number of vehicles over its entire trip. This information is crucial as it provides details on its ability to sustain the rate of updates over time.

This information is also useful in making decision for selection and features of routing protocol. Some dissemination protocols collect information from its 1 hop neighbor to determine the routing decisions, such as one that gathers the list of 1 hop neighbors and then based on the unique set of neighbors, one amongst the neighbors of the current broadcasting node is made the next broadcaster. Some utilize neighborhood discovery to maintain and create links between vehicles. For all such protocols, the performance of the algorithm depends on the average number of neighbors one has over the entire time duration. We can see that the performance of the system heavily depends on number of neighbors as time required to exchange information has at least an order of $\Theta(n)$. Finding how frequent n vehicles are in neighborhood would help in deciding the practicality of dissemination protocols.

Analysis is conducted to calculate the number of instances of simultaneous communication for different settings of link availability times (1/5/10 seconds) with Communication range of 350 m. The results are shown in Table 4-4 while a graphical representation is shown in Figure 4-10.

Time	>=1	>=2	>=3	>=4	>=5
1 sec	59773136	11713501	7393523	3942305	3116892
5 sec	10573672	2096664	1279938	730381	567579
10 sec	4548549	958078	573790	344369	266695

Table 4-4: Number of Instances of Simultaneous Connections w.r.t Link Availability Time

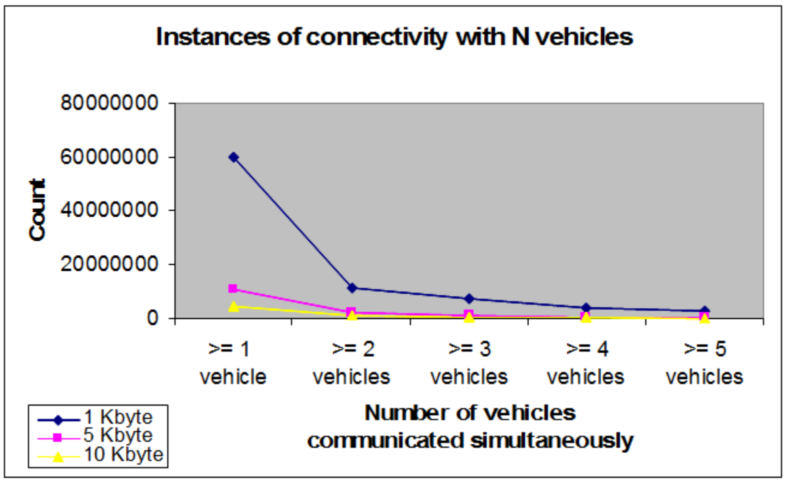


Figure 4-9: Number of Instances of Simultaneous Connections

The results reveals that VANETs tend to form clusters on numerous instances over the entire day. The frequency of clusters of vehicles decreases as link availability for communication increases. These results imply that there will be many instances where number of nodes are more than two and hence there will be overhead involving 2-way communication or make use of neighbor discovery impacting the ability to spread the update to other vehicles. Considering an example from the table where, if the overhead for communication between the vehicles takes about 1 s while the data requires about 3 s, the chances of communicating with N neighbors reduces by approximately 20% and hence we require a protocol that can operatre with minimum possible amount of overhead of communicating with neighbors.

4-11 Summary

In this chapter we were able to provide key insights into mobility and communication capabilities of traffic in the Netherlands (based on the samples collected). The information gathered such as the variations of traffic over 24 hours, rate of new vehicle activations, churn in VANET and others will be used as inputs for design of the system and routing protocol for spreading updates. We found the following parameters interesting.

Dis-connectivity:
This parameter revealed that, there are always vehicles which will never communicate with any other vehicle. This number provides us with the upper limit that can be achieved with any dissemination protocol deployed over this VANET. It also acts as a comparison parameter to analyze the performance of various dissemination protocols where one can compare actual routing performance without having to consider the shortcomings associated with traffic.

Connectivity:
The term connectivity determines the possibility of vehicles to communicate or update other vehicles over its entire trip. This parameter responds negatively to any increase in the link availability time. The analysis revealed that the PDF of the parameter is best described as exponential distribution implying that possibility of finding vehicles which update other vehicles reduces as the number of updates increases. Hence it is nearly improbable to find a set of vehicles which can solely be made responsible for updating the entire traffic

Degree of Connectivity:
The term degree of connectivity determines the ability of vehicles to update multiple vehicles at the same instant. This parameter responds negatively to any increase in the link availability time. The analysis revealed that the PDF of the parameter is best described as exponential distribution implying that possibility of finding vehicles which update multiple vehicles simultaneously reduces as the number of vehicles to update increases. Hence VANETs are more likely to update individual vehicles rather than more number of vehicles. Thus in order to spread updates amongst the vehicles the only way is to have larger fraction of traffic as source of updates rather than relying on small subset to achieving the required spreading of updates.

Chapter 5

Simulator

Analysis and validation of a network is best conducted by use of the real world testing, while situations where this is not feasible, simulators are utilized. VANET, a network of vehicles based on IEEE 802.11p technology is in early stages of development and so hardly any commercial products supporting IEEE 802.11p wireless communication are available making the communication kits expensive. The support of infrastructure in form of Road Side Unit (RSU) is also far too less with only few existing in test beds around the world making any form of testing only restricted to these small areas. Above all, the cost in terms of time and money associated with real world testing makes it impractical to have numerous test runs. Therefore, various different simulators are used to validate one‘s research work in VANETs. Over the years various simulators have propped up and have been made available, with few such as NS2 gained popularity and acceptance amongst the research and academic world. These simulators work well for most of applications but for certain category of applications such as ours they end up being unsuitable requirements of our application bring forth the shortcomings of the tool making it unsuitable for use for our application. To overcome these shortcomings, we propose a new simulator named DC-PIC. This chapter discusses the shortcomings, followed by the concept of our new simulator. The chapter ends with discussion on the implementation of this simulator.

5-1 Problem Size

Most simulators rely on mobility models and traffic simulators to provide mobility for the models we work with real world non intrusive GPS data providing the mobility patterns of the vehicles. This data is collected by TomTom International B.V from its PND devices for its various services. The data collected adheres to the privacy policy of the customers and hence all the information that could trace back to the customer is filtered out before used for any purposes. There is no trace of the individual vehicles or their owners. Further, the data is a few months old and thus can not impact any decision making at this time. The data used in this thesis is for the purpose of post analysis and learning the behavior. Further, the work in this thesis tries to provide a framework which could be extended to work with real-time data assuming the privacy policies are adhered to and authorities allow this in future.

```
Trace Data  = { TimeStamp, Latitude, Longitude }
```

The data collected is stored as a triplet of timestamp, latitude and longitude values for every time interval during the time PND device is active. Average trips runs in tens of minutes while the values of timestamps are in seconds. This ends up with lot of traces collected for each trip a vehicle undertakes. With time period of 24 and area as big as the Netherlands we have large number of vehicles performing different set of trips presenting us with a huge data to process. The traffic data for mobility collected for 1st August 2011 has 94,000 vehicles with collective size of traces approximating to 1.2 Gb.

5-2 Issues with Current Simulators

Various simulators have been developed over the years of which few were discussed in Chapter 1. Of all these, ns2 is most popular and hence we will concentrate on feasibility of this tool to utilize this large set of traffic data for simulation study.

ns2 an open source tool is considered as a de-facto simulator for adhoc networks. It is utilized for VANETs with the help of support of traffic simulators which provide mobility patterns for vehicles in VANETs. Having contributions and support from vast research communities from all over the world it provides extensive support of protocols across various communication layers. The modular structure of code and ease of modification are few of its advantages that make it favourable to work with for different studies. Yet it has few of the shortcomings which fail it to be utilized for our application of map updates. We analyze it this section.

5-2-1 Memory Issues

```
$ns_ at 120 "$node_(9) setdest 5008.43 1423.59 14.94"
```

Concerns of memory requirements made the trace data collected be stored in a non-human readable format while the ns2 traces require format which are to be human readable. Hence we need to convert already a large set of traces into human readable format making the memory utilized by the traces after conversion to balloon up to 5Gb. Apart from the conversion ns2 requires additional information be present along with each trace to that required by the simulator which balloons up the memory required by the traces to exceed 8Gb. Such memory sizes are can be accommodated in secondary memory devices such as hard disks. But with ns2 where it requires all the mobility traces be read into primary memory (RAM) makes ns2 inoperable as even a high end desktop can boost of 8Gb of RAM just for application. An attempt to run it with ns2 makes the Ubuntu OS on Intel i7 processor desktop aborts the execution of ns2 application as soon as memory consumed by the application approaches the limits of its RAM.

5-2-2 Number of Nodes

$$n * (n-1) * (t/b) \tag{5-1}$$

In ns2 the ability of nodes to receive the transmitted data from one of the nodes is based on calculating the received power at each and every node in the network. It is compared with a *Threshold* value which decides whether to accept or reject the packet. This mechanism works very well when dealing with nodes up to a few thousands but when working with huge set of nodes such as our setup this mechanism is too slow to perform any simulation. Consider a network of n be number of nodes that are active for time period of t and for a simple flooding algorithm which has broadcasts every b s one would have to calculate the received signal the number of times as indicated in equation Eq.(6-1) having a run time complexity of $\Theta(n^2)$. For a system of even 10,000 nodes running for 24 hours with broadcast at every 5 seconds would end up having 1727827200000 computations of received power! This implies that even if we were able to resolve the previous issue of memory we would still have issues with the implementation of the simulator being not at all scalable with respect to number of nodes.

5-2-3 Properties of Nodes

When dealing with time of simulation as long as 24 hours, the number of vehicles running at any given moment of time are never constant but continuously changing as seen in the previous chapter. This feature of having nodes being active or inactive during simulation is inherently unavailable in ns2 and hence one would spend time computing existence of anode in its proximity even if it is no longer running in transmitting node‘s neighbourhood. Even though one can add the code to able to do the same it is still inherently unavailable and hence makes it a limitation for ns2. Issues discussed for ns2 also hold for other simulators as none has been developed with an objective to operate on real world traces of such magnitude. These limitations make none of the existing simulators operable to analyze map updates as an application. Memory and computation time being the major issues associated with the current simulators we designed our simulator so as to overcome problems associated with them.

5-3 DC-PIC VANET Simulator

DC-PIC VANET Simulator is the simulator we propose for handling real world mobility traces to analyze Map Updates as an application. The term "DC-PIC" is an acronym for **D**ivide and **C**onquer and **P**article **I**n **C**ell Method. These 2 methods which are extensively used across in High Performance Computing in various fields such as plasma physics, heat transfer are the basis on which the simulator is developed.

5-3-1 Computation Time

The issue of computation time arises due to the fact that we need to compute received power (R_x) at each vehicle in the VANET, and with the number of vehicles exceeding the tens and thousands makes the simulation time to be so long that one cannot use it simply for any analysis. One of the approaches would be to use supercomputers which provide vast computational power to resolve this. But as supercomputer is not cheap and neither easily available, adaptation of the tool would be hindered amongst the research community.

Another solution is to make use of some high performance computing techniques which can reduce the number of computations so that one can work with the tool on the standard desktop machine. These techniques of reduction of computations come at the cost of loss of precision. The acceptable loss of precision will decide the factor by which computations are reduced. The techniques used are discussed in the following subsections.

Divide and Conquer

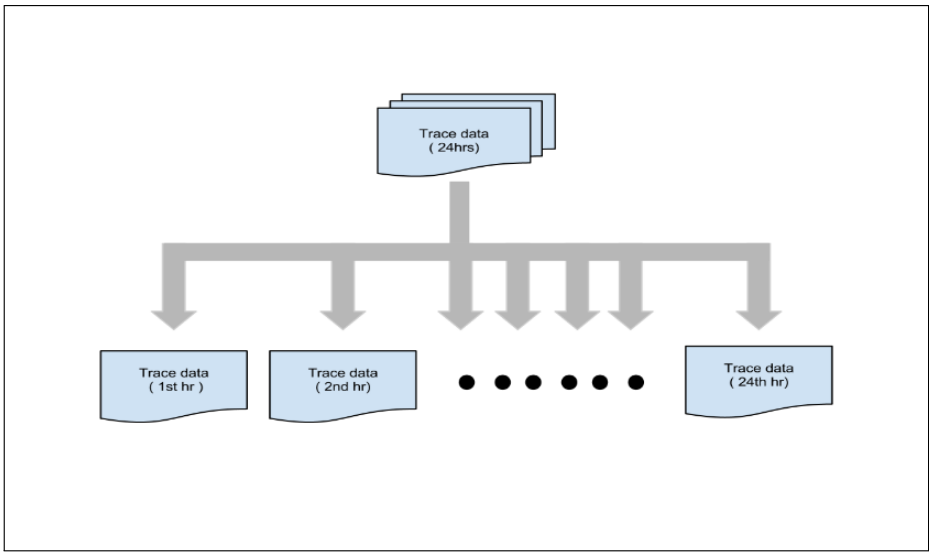


Figure 5-1: Divide Data Set based on Time

The technique of divide & conquer, utilized in various fields of life is a mechanism by which the entire complex problem is resolved by breaking it into smaller problems and operating on them sequentially or in parallel manner. In our application, the idea is to segregate traffic into smaller sets, such that when operating on an element only has to compute received power among the other elements of the same data set rather than the entire traffic. Hence, if segregation results in s average number of vehicles per N_{sets} number of sets and with a relation with total number of vehicles as $s = n/N_{sets}$, the Eq.(6-1) gets updated to as shown in Eq.(5-2) with reduced the run time complexity of $\Theta(N_{sets} * (s^2)$ from $\Theta(n^2)$ where $s << n$.

$$\frac{n}{N_{sets}} * (\frac{n}{N_{sets}} - 1) * \frac{t}{b} * N_{sets} \tag{5-2}$$

Eq.(5-2) clearly shows that the performance improvement lies s which is the ability to segregate the entire set into smaller sets. Each of the three parameters, time, latitude and longitude, can be used to segregate into smaller sets. Use of latitude and longitude, either alone or in combination is not beneficial because of the practical difficulties associated with it. They are, first is that if one segregates the data set into smaller blocks to avoid computation of received power, the number of sets generated is very large, for example country like Netherlands would require $860X860$ sets for communication rage of 350 meters. This large number of N_{sets} too large for manipulation and computation for further steps and hence dampens the performance improvement .Whereas, if sizes are made larger to maintain N_{sets} in acceptable range, the number of vehicles per set would increase making $s >> 0$ and again dampening the performance improvement. Therefore, we decided against the use of latitude and longitude as a parameter for divide and conqueror method.

Instead Time, present in form of timestamp in traces, acts as a natural filtering agent by segregating vehicles which travel at different non overlapping times. It is an excellent filtering agent because vehicles active at different times in all aspects do not require interacting with each other. Neither time duration needs to be that small so as to create issues which are with use of latitude and longitude. Using time as a parameter, we end up with equation as shown in equation Eq.(5-3) where t_{new} is the new time duration for each set is created. An example of the split is shown in Figure 5-3-1 where the entire data of 24 Hrs is split into 24 files each for one hour.

$$\frac{n}{K_f} * (\frac{n}{K_f} - 1) * \frac{t_{new}}{b} * N_{sets} \tag{5-3}$$

The resolution of parameter t_{new} , decides the number of sets created. As discussed previously, larger the number, better the performance can be achieved. But increasing it too high makes the sets very similar as compared to its neighboring sets. This is because; vehicles even though being mobile in nature have a nominal speed limit due to various physical and social constraints. Hence vehicle mobility data is segregated for higher resolution like 1 second, we would end up with minimal changes to the data set between the sets representing consecutive time values. On the other hand, if resolution is kept low $t_{new} \to t$ we end up with $s \to n$ and so loosing any advantages from using this technique. Hence selecting a right value is crucial. For our analysis we considered a resolution of 1 minute as it is neither small to create redundancy issues nor large enough to result in having too many vehicles in a set to diminish with performance improvement obtained with divide and conquer approach.

Divide and conquer approach reduces the number of vehicles to process n/N_{sets} but the need of calculating received power (R_x) still remains. However, large number of sets are created one would always need to perform received power calculations with all the vehicles part of the same set. Therefore, to resolve this issue we propose to use the technique known as Particle In Cell method

Particle In Cell

Particle In Cell method is a technique primarily used in high performance computing in domains such as computations of gravitational forces in the universe. The strategy involves overlaying a grid over the entire region under simulation and then approximating the particles in simulation to a single point to a nearest grid point so that improve the simulation time with least amount of loss of precision. This section explains the use of this strategy in our simulator.

$$P_r = P_t * G_t * G_r * (\frac{\lambda}{4 * \pi * R})^2 \tag{5-4}$$

Friis transmission model as shown in Eq.(5-3-1) is widely used as the propagation model in wireless communications. The equation calculates received signal power, P_r, based on Transmitter Power P_t, Gains of transmitting and receiving antennas G_t and G_r respectively, and wavelength λ and distance between the transmitter and receiver R. With all parameters apart from the distance being constant we can conclude that the ability of receiver to receive message from he transmitter solely lies depends on the distance between these vehicles as shown in Eq.(5-3-1).

$$P_r \propto (\frac{1}{R})^2 \tag{5-5}$$

If the maximum distance at which a transmission can be received from a transmitter is defined as $R_{threshold}$ then for all potential vehicles who have distance from transmitter R which satisfy $R \leq R_{threshold}$ would receive the transmission while the ones which fail the criteria would not. Thus we propose formation of a grid of squares of dimension of $R_{threshold}$, so that vehicles which fall in a particular square region can be assumed to be within the communication range without actually requiring to compute received power. Whereas, all the vehicles that fall outside the given square region can be discarded for calculating received power as they would never receive the power because $R > R_{threshold}$. Using $R_{threshold}$ as a dimension of the squares a grid is formed over the area under simulation with an example shown in 5-3-1

Divide and conquer approach reduced the number of vehicles for computations but even after reduction the number of vehicles remained quite high, making the simulation time not yet fast enough to be practical in nature. In our data set at the highest traffic

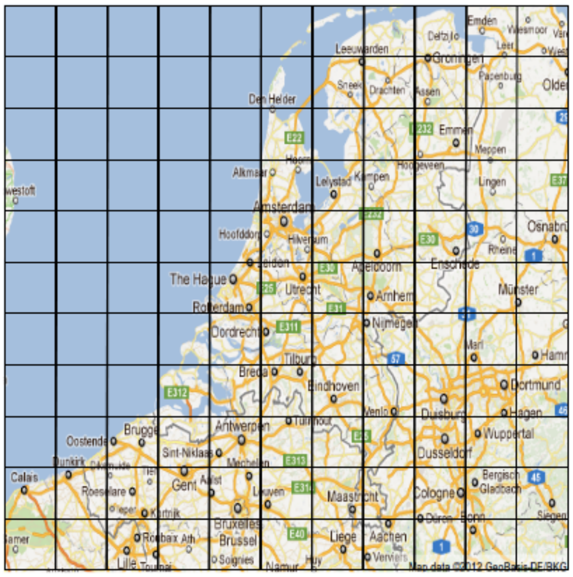


Figure 5-2: Grid on Netherlands

load the number of vehicles is around 27,000 nearly $\frac{1}{3^{rd}}$ of overall count of vehicles, which is still quite high. The overlay of grid enables to reduce the number of vehicles in consideration for computation to reduce further. After overlay of grid we have g_t where $g_t = s/K_d$ as the average number of vehicles in a grid to be considered for communication of received power in the grid of $N_{squares}$ square, the equation Eq.(5-3) gets modified as shown in equation Eq.(5-6) with run time complexity of simulator as $\Theta(N_{sets} * N_{squares} * (\frac{n}{K_d.K_f})^2)$.

$$\frac{n}{K_d.K_f} * (\frac{n}{K_d.K_f} - 1) * \frac{t_{new}}{b} * N_{sets} * N_{squares} \tag{5-6}$$

In spite of all the work one still needs to perform computations for every transmitter and as the number of transmitters increase so does the computation time. As already stated that all the vehicles inside a particular square region would be able to communicate with each other the computation of received power is unnecessary and hence we can avoid it all together by assigning all the vehicles within the square a new pair of latitude and longitude values – a unique pair for each of the square regions in a grid. This is known as aggregation of vehicles. It makes us no longer needing to compute the received power but only compare the new values of the locations associated with vehicles so as to find whether the vehicles exist within each others proximity to be able to communicate with each other. Moving from computations to comparisons is a huge benefit as it is less CPU intensive providing excellent improvement in simulation time. An example of this approach is depicted in Figure 5-3-1.

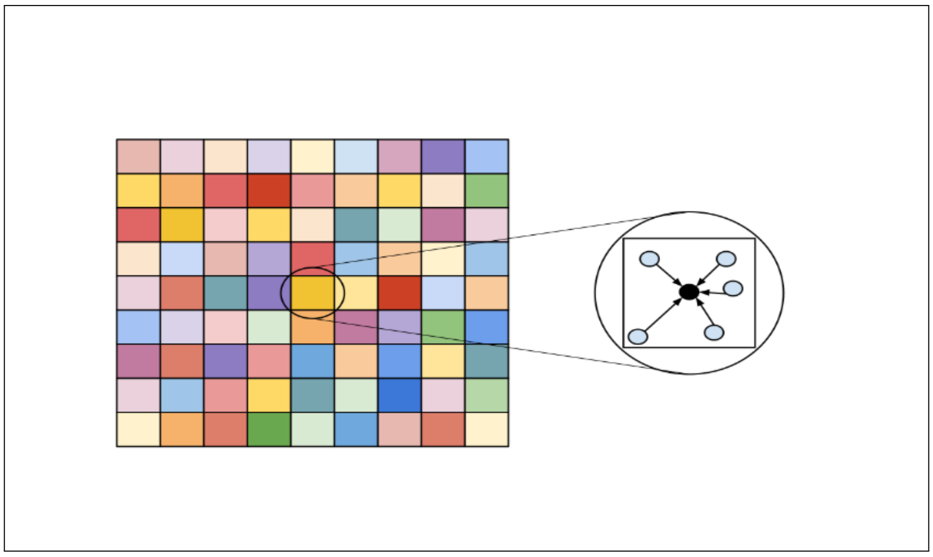


Figure 5-3: Approximation of Locations

Issues

The process of aggregation discussed before has the capabilities to provide an excellent improvement in simulation, but it comes at the cost of loss of precision. The loss of precision occurs in scenarios where vehicles, even if they have their distance R satisfying the requirement of $R \leq R_{threshold}$, cannot be considered to be able to communicate with each other due to different allocation of squares in the grid. This issue arises due to the fact that the grid is generated with a fixed origin which in our case is left hand bottom corner of the map under consideration whereas in wireless communication the distance of potential vehicles is measured with the transmitter as the origin. An example of this is depicted in Figure 5-3-1 where even if Vehicles C even if shorter in distance to Vehicles A and Vehicles B than the distance between Vehicles A and Vehicles B, a valid communication link will not exist between Vehicles C with either of Vehicles A or B.

Solution to generate grid with every transmitter and then assign vehicles to the respective grids for every time interval is not a feasible solution so one needs to look for a different strategy to resolve this problem.

Solution

The problem discussed in the previous section occurs due to the lack of knowledge of vehicles associated with neighbouring grids and to resolve it one must exchange the set of vehicles associated with grids with all its 9 surrounding neighbours which are within the communication range distance but fall in separate grids. But as we have aggregated all the vehicles within a grid onto a set of vehicles at one single location, there is no way for neighbouring grids to know the exact location of these vehicles and would result in error calculating the set of potential communication links. Thus to resolve this grids of finer resolution are placed above the coarser grids so that one can aggregate location of

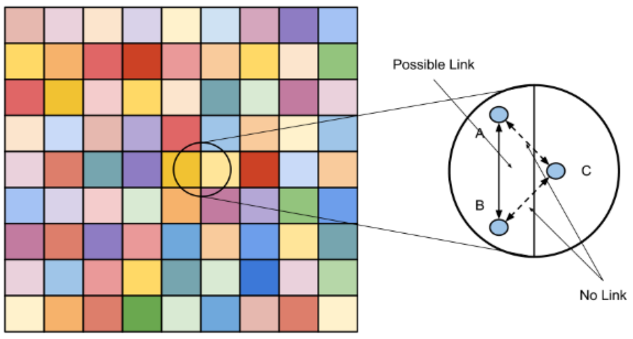


Figure 5-4: Error due to Grid Formation

vehicles over much smaller areas providing the accurate set of vehicles to be exchanged with its neighbours. This system works but at the cost of increase in communication and data exchange time. For a finer grid of nXn the maximum exchange of data from any of the grids is $5 * n * (n - 1)$. This makes the system not scalable with higher resolution of finer grids. An example is shown in Figure 5-3-1 below where parent grid is of 500mX500m which comprises of 4 finer grids each of 250mX250m in size depicting the flow of exchange of set of vehicles from one coarse grid to its neighbor grids.

Solution to this problem is to propose an additional set of coarse grids with fixed offset between them. The idea is that number of coarse grids decides the accuracy of data. These set of coarse grids are placed on top of existing Grid but with an additional offset. Such a setup makes it possible for pair of vehicles which are within the communication range to be in the same square in one of the coarse grids. When such a scenario occurs between a pair of vehicles a valid link between these vehicles is generated. Therefore as the number of coarse grids increase more and more accurate communication links can be setup. An example of 4 overlapping grids is shown in Figure 5-3-1.

5-3-2 Memory

The issue of memory is resolved by the proposal to make use of files as means of storing the intermediate as well as the final results in form of a file. This helps to take the advantage of abundant secondary memory storage available on our standard desktops. The advantage comes at the cost of overhead of storing and reading from the memory at every intermediate steps of execution. Considering worst case time to access file in secondary storage is 12ms, one needs $2 * n * 12$ ms to read and write into the memory. This reduction in speed of operation is necessary to overcome the limitations of sizes of primary memory.

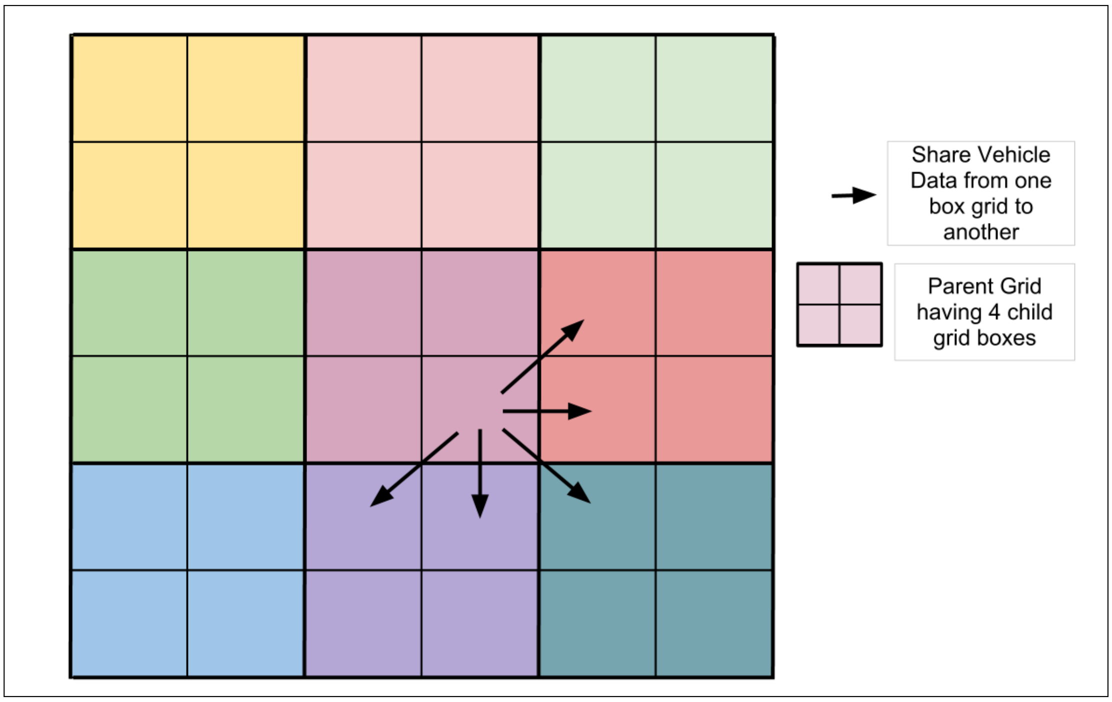

Share Vehicle Data from one box grid to another
Parent Grid having 4 child grid boxes

Figure 5-5: Exchange of vehicle Data with neighboring grids

5-4 Implementation

Simulator is implemented as two loosely coupled modules with one handling the mobility analysis while the other is responsive for network simulation. The module of mobility analysis handles the GPS trace data for the traffic and generates the potential communication links between two vehicles. This is stored onto a file which is later then provided as an input to the network simulation where the actual application of spread of map updates is implemented and tested.

The purpose of following such a design is that by doing so as once the potential communication links are created there is no need to generate them again in subsequent simulation runs until one changes the communication parameters such as communication range. Further this process helps to provide ease of integrating any modifications associated with the RSUs without the need of re-executing mobility analysis module. Similarly having network simulation decoupled from the computationally expensive mobility analysis module makes development or modifications of any protocols very easy to integrate and verify its performance.

5-4-1 Mobility Analysis

1. **Interpolate Trace Data** : This step is crucial as the trace data which is collected as part of GPS traces are collected via different sources and hence have different sampling rate. Hence all the vehicles do not provide latitude and longitude values at the same fixed interval. And in order to standardize all the trace data into

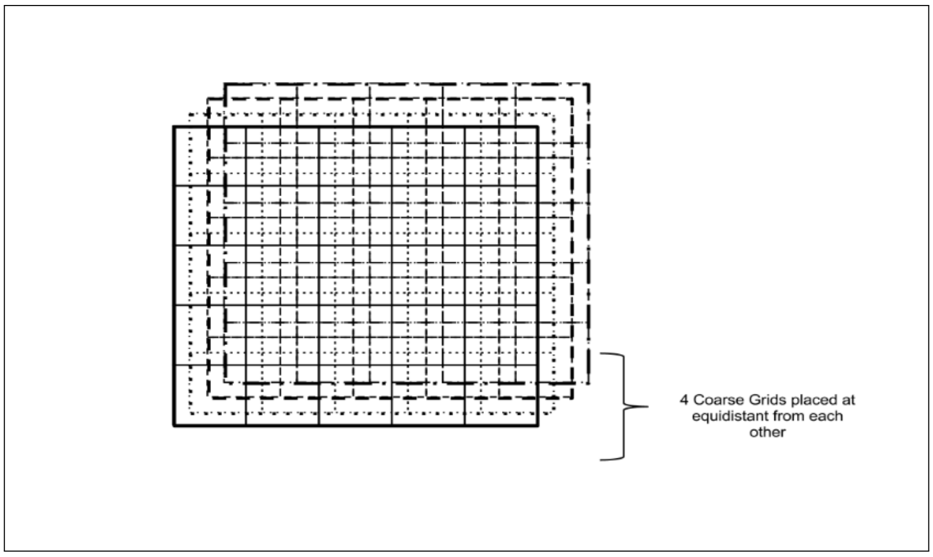


Figure 5-6: Overlapping Grids

one single sampling period the intermediate values are interpolated. Apart from difference in sampling rates there are also issues of missing traces either due to GPS signal loss or either the Vehicle had switched off at a certain location for a small certain duration. In the latter case interpolation would result in generating vehicle movement even when it was not actually running and hence to avoid filtering such cases a time gap is measured and in case it falls above a certain threshold (we use five minutes as threshold).

2. **Generate Grid Parameters**: Based on the latitude and longitude extremities of region under consideration the grid sizes and associated parameters are calculated. The purpose of calculation of the grid is required as direct mapping a square region of a map based on latitude and longitude when projected on Earth‘s surface is no longer a square resulting in incorrect implementation of the simulator. The calculation is conducted using trigonometric methods and the algorithm is mentioned in Algorithm 5-6.

3. **Assign latitude and longitude identifiers**: As discussed before the parameters of latitude and longitude are stored in floating point format and hence one would require huge set of floating point operations in order to find the distances between the vehicles. This makes the simulation extremely slow process and hence we map the individual locations to a region of 10mX10m region so from hence forth comparison of individual latitude and longitude identifiers would suffice to determine their proximity and no longer need computationally expensive floating point operations to execute. As comparison is far less cpu intensive that floating point operation one gets immediate improvement in speed of simulation.

4. **Assign Coarse and Fine Grids**: The new values of Latitude and Longitude for every vehicle are then used to assign vehicles with coarse and fine grids. The algorithm 5-6 explains the execution of this step.

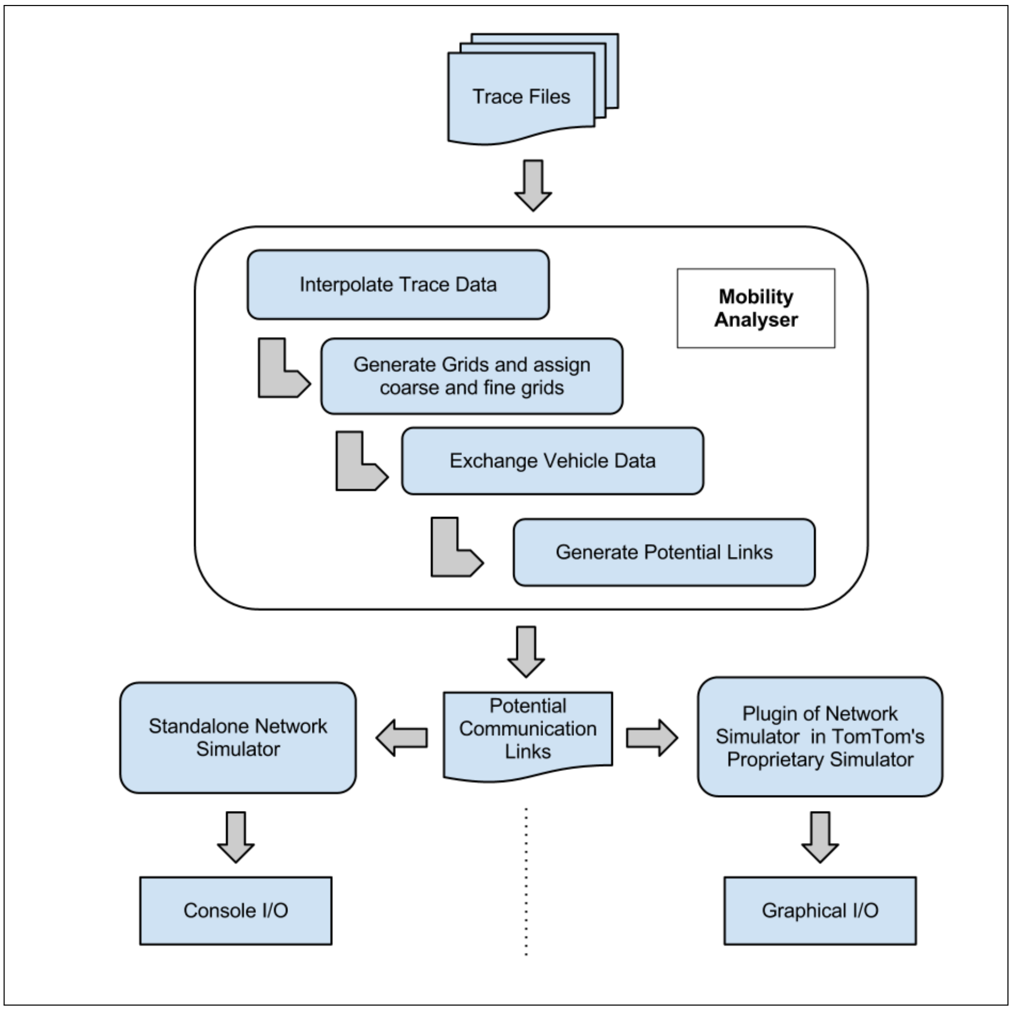

Trace Files
Interpolate Trace Data
Mobility Analyser
Generate Grids and assign coarse and fine grids
Exchange Vehicle Data
Generate Potential Links
Standalone Network Simulator
Potential Communication Links
Plugin of Network Simulator in TomTom's Proprietary Simulator
Console I/O
Graphical I/O

Figure 5-7: Block Diagram of Simulator

5. **Assign Coarse and Fine Grids**: The new values of latitude and longitude for every vehicle are then used to assign vehicles with coarse and fine grids. The Algorithm 5-6 explains the execution of this step.

6. **Link Generation**: This is the final step of execution where the actual links are generated and stored in one single file. The Algorithm 5-6 explains the implementation of this step in a simplest manner.

The network simulator involves using the communication links generated in mobility analysis module to decide on, which vehicle in the proximity would get the update. The module is built with every vehicle built as an individual object consisting of an application layer where parameters such as map update type and size, Current location and trip data such as start or end destination is stored. This module also stores the information about the type of update it received which is whether it has received the entire map update or is partially updated. If updated then the source of update is also stored. Network Layer is where the segmentation of map to be broadcasted or merging

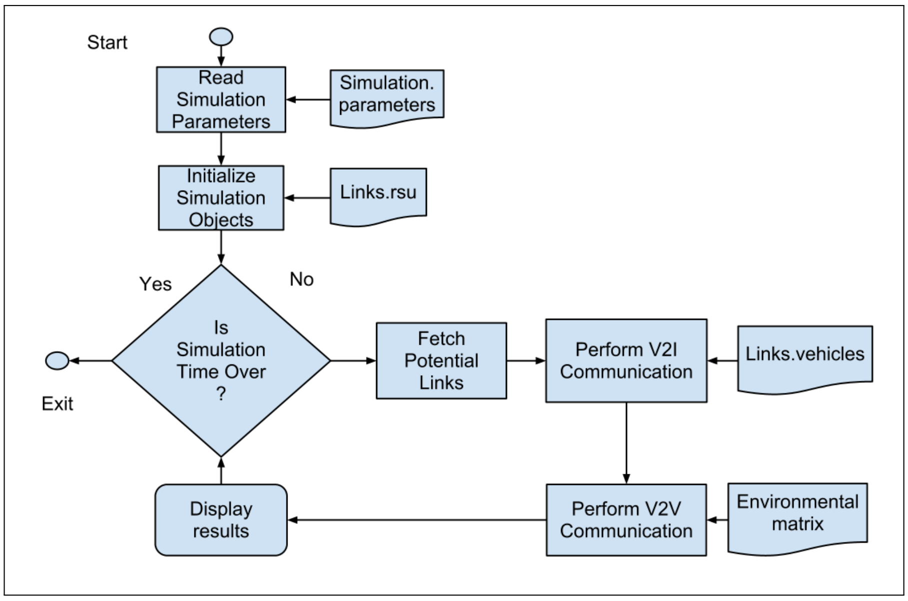

Start
Read Simulation Parameters
Simulation. parameters
Initialize Simulation Objects
Links.rsu
Yes
No
Is Simulation Time Over ?
Exit
Fetch Potential Links
Perform V2I Communication
Links.vehicles
Display results
Perform V2V Communication
Environmental matrix

Figure 5-8: Flowchart of Network Simulator

of various map segments received is executed. It also stores information related to communication such as number of Broadcasts, number of Redundant and Collision messages. It makes use of these parameters and parameters from application layer to decide whether to broadcast the update or not. MAC layer is responsible for performing random back off, checking for collision of messages and reporting these issues to upper layer. The Physical Layer with the help of communication link file decides whom the vehicle interacted with. It also provides a feature of adding a probability factor which can be used to simulate physical constraints. The flowchart is depicted in Figure 5-4-1 while Figure 5-4-1 depicts communication model of the vehicle. This model is built on the lines of the model used in ns2. The design of network simulator is inspired from ns2 so as to be able replicate the advantages of management of software modules, ease of modification and implementation of any new topics.

Plug-In for TomTom‘s Simulator

The network simulator comes in two versions: standalone, which is mainly used for network analysis and a plug-in for TomTom‘s internal simulator. This simulator has the capability of graphical interface showcasing the movement of traffic on the NDS Digital Maps. The purpose of the plug-in is to enable a graphical representation of spread of updates in the real world. A snapshot of the working of simulator along with the plug-in is shown in Figure 5-4-1. Key features of features of this plug-in include:

- A graphical view of the status of updates amongst the traffic where vehicles are with red colour indicates *Not updated*, blue indicates *update from RSU* while green indicates *updates from other vehicles.*

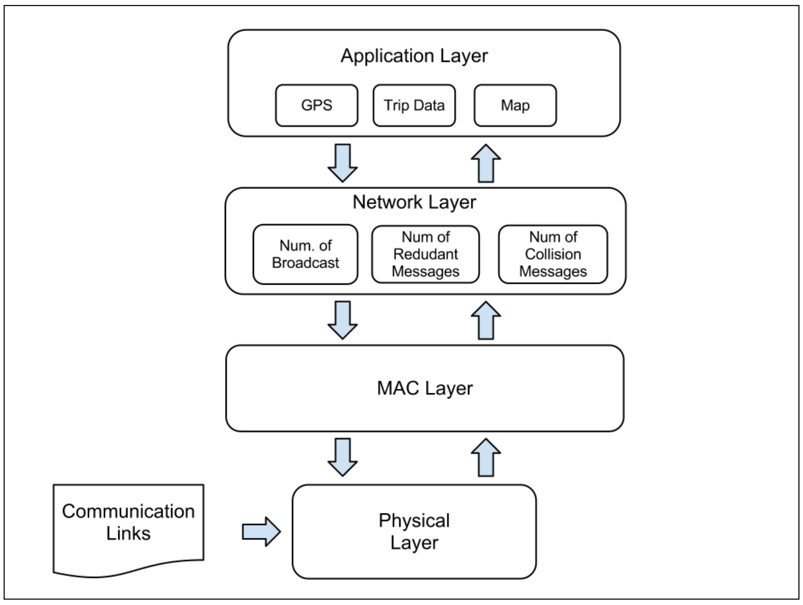


Figure 5-9: Communication model of Vehicle

- Run time feature to update vehicles with the help of one single click of mouse button.
- An indicator showcasing the current status of traffic updates.

5-4-2 Simulation of Environmental Hindrances

Apart from the obvious packet loss in during wireless communication due to attenuation, and fading of signal, surroundings will induce additional losses reducing the communication range further. This being very specific to surroundings, it is very difficult to have any mathematical model to generalize the same. The overlay of grid structure on top of actual map enables us to setup individual probability of availability of a link for each of the square regions in the grid. This value is used to state that with this probability the link at that location exists. For example cities due to high structures tend to have smaller communication range capabilities as compared to highways in open areas and therefore we can assign a lower probability to the regions encompassing the cities as compared to other regions to indicate the lower possibility to communicate in cities as compared to open areas. The blocks are depicted as, Environmental Probability Matrix in Figure 5-4-1.

5-5 Summary

The available simulators lack support for use of large set of trace data making us to develop our very own simulator based on techniques used in High Performance Computing. This simulator will be further investigation of Map Updates as an application

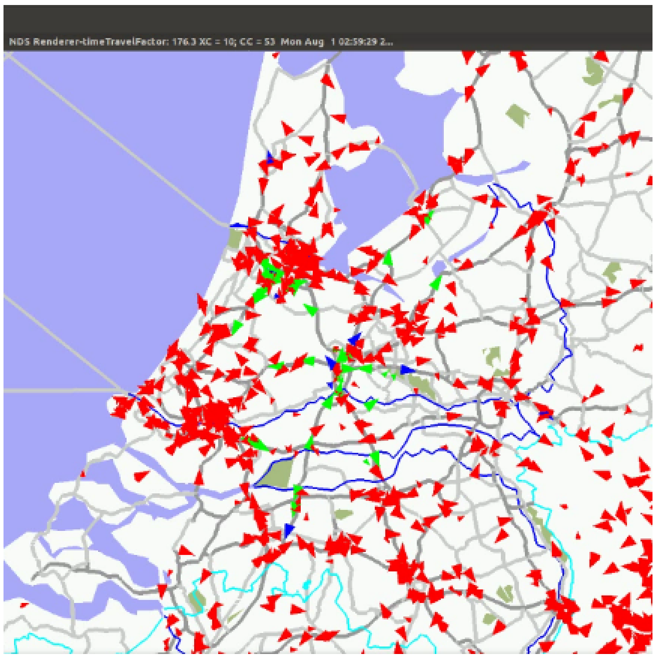

NDS Renderer-timeTravelFactor: 176.3 XC = 10; CC = 53 Mon Aug 1 02:59:29 2...

Figure 5-10: Communication model of Vehicle

for VANETs. Being generic in nature, it can be used to test any VANET application and not just Map Updates.

5-6 Algorithms

Algorithm 1 Interpolate Trace Data

```
Require: $ReStartTime = 30$
  for all $Vehicles$ do
    $prevTime \leftarrow 0$
    $prevLatId \leftarrow 0$
    $prevLatId \leftarrow 0$
    for all $TraceData$ do
      $currTime \leftarrow fetchTimeFromFile()$
      $currLatId \leftarrow fetchLatFromFile()$
      $currLatId \leftarrow fetchLngFromFile()$
      if $currTime - prevTime > 1$ then
        if $Time - prevTime < ReStartTime$ then
          $timeDiff \leftarrow Time - prevTime$
          $latDiff \leftarrow currLatId - prevLatId$
          $lngDiff \leftarrow currLngId - prevLngId$
          for $count < timeDiff$ do
            $newTimeValue \leftarrow (prevTime + count)$
            $newLatValue \leftarrow prevLatId + count * (latDiff/timeDiff)$
            $newLngValue \leftarrow prevLngId + count * (lngDiff/timeDiff)$
          end for
        end if
      end if
      $prevTime \leftarrow currTime$
      $prevLatId \leftarrow currLatId$
      $prevLatId \leftarrow currLatId$
    end for
  end for
```

Algorithm 2 Distance between 2 points (distanceCalculation())

```
Require: $latValue1 \geq 0, lngValue1 \geq 0, latValue2 \geq 0, lngValue2 \geq 0, earthRadius = 3958.75, meterConversion = 1609$
  $latDiff \leftarrow Radians(latValue2 - latValue1)$
  $lngDiff \leftarrow Radians(lngValue2 - lngValue1)$
  $lat1Rad \leftarrow Radians(latValue1)$
  $lat2Rad \leftarrow Radians(latValue2)$
  $temp \leftarrow \sin^2(latDiff/2) + \sin^2(lngDiff/2)^2 . \cos(lat1Rad) . \cos(lat2Rad)$
  $result \leftarrow 2 * meterConversion * earthRadius * \arctan(temp^{1/2}, (1 - temp)^{1/2})$
  return $result$
```

Algorithm 3 Assign Coarse and Fine Grids (assignGrids())

```
Require: $latMax \geq 0, latMin \geq 0, lngMax \geq 0, lngMin \geq 0, size = 10, range = 50, resl = 2$
  $latIndexes \leftarrow distanceCalculation(latMax, 0, latMin, 0)/size$
  $lngIndexes \leftarrow distanceCalculation(0, lngMax, 0, lngMin)/size$
  $latGrifOffset \leftarrow (latMax - latMin)/latIndexes$
  $lngGrifOffset \leftarrow (lngMax - lngMin)/lngIndexes$
  for all $Vehicles$ do
    for all $Time$ do
      $Vehicle.latId \leftarrow (fetchLatFromFile() - latMin)/latGrifOffset$
      $Vehicle.lngId \leftarrow (fetchLngFromFile() - lngMin)/lngGrifOffset$
      $CoarseGrid[int(Vehicle.lngId/range)][int(Vehicle.latId/range)] \leftarrow Vehicle$
      $Vehicle.FineLngId \leftarrow int(Vehicle.lngId/resl * range)$
      $Vehicle.FineLatId \leftarrow int(Vehicle.latId/resl * range)$
    end for
  end for
```

Algorithm 4 Exchange and Create Links (createLinks())

```
Require: $latValue1 \geq 0, lngValue1 \geq 0, latValue2 \geq 0, lngValue2 \geq 0, resl \geq 0$
  for all $Time$ do
    for all $latId \rightarrow Latitude$ do
      for all $lngId \rightarrow Longitude$ do
        $CoarseVehicleArray[] \rightarrow CoarseGrid[lngId][latId].vehicles$
        for all $c \rightarrow CoarseVehicleArray[]$ do
          $FineVehicleArray[(resl * c.FineLngId) + FineLatId] \leftarrow c$
        end for
        $tempArray[] \rightarrow CoarseGrid[lngId - 1][latId].vehicles$
        $tempArray[] \rightarrow CoarseGrid[lngId - 1][latId - 1].vehicles$
        $tempArray[] \rightarrow CoarseGrid[lngId][latId - 1].vehicles$
        $tempArray[] \rightarrow CoarseGrid[lngId][latId + 1].vehicles$
        $tempArray[] \rightarrow CoarseGrid[lngId + 1][latId + 1].vehicles$
        $tempArray[] \rightarrow CoarseGrid[lngId + 1][latId + 1].vehicles$
        for all $c1 \rightarrow CoarseVehicleArray[]$ do
          $FineVehicleArray[(resl * c1.FineLngId) + FineLatId] \leftarrow c1$
          for all $c2 \rightarrow tempArray[]$ do
            if $|c1.FineLngId - c2.FineLngId| \leq resl$ then
              if $|c1.FineLatId - c2.FineLatId| \leq resl$ then
                $FineVehicleArray[c.FineGrid] \leftarrow c$
              end if
            end if
          end for
        end for
        for all $CoarseVehicleArray \leftarrow c1$ do
          for all $CoarseVehicleArray \leftarrow c2$ do
            if $c1 \neq c2$ then
              $links \leftarrow time, c1, c2$
            end if
          end for
        end for
        for all $FineVehicleArray \leftarrow c1$ do
          for all $FineVehicleArray \leftarrow c2$ do
            if $c1 \neq c2$ then
              $links \leftarrow time, c1, c2$
            end if
          end for
        end for
      end for
    end for
  end for
```

Chapter 6

Source of Updates

Map Generation is complex process of collection, validation and compilation. This process due to the technical and commercial aspects is always done at central server which is outside the VANET network. There is always a reliance on outside network to provide VANET with the changes. This is accomplished using either the Road Side Unit (RSU)(permanent / temporary) or the vehicles themselves as sources of updates. This chapter basically compares these types of sources of updates and discusses how parameters of these sources influence spread of updates.

6-1 Map Updates, an Analytical Equation

The map update application in VANETs will have a vehicle updated with new information from either primary source or a secondary source. We define a primary source as one which has received its update from outside that of VANET while the secondary source as one which received its update from one of the primary sources. As with most other applications, RSU will be predominantly the primary sources in VANETs which receive their updates from the servers of map supplier while vehicles can be primary sources of updates if it has received its update via an external mechanism such as home/office Wi-Fi network and as a secondary source when it has received its update form RSUs or other vehicles.

The ability of RSUs to update vehicles is primarily influenced by the location at which the units are placed. The location is a vital parameter as it determines the traffic pattern at any location. For example, regions near residential and commercial complexes could end up having traffic pattern showing peaks around start and end of office hours where as locations such as highways would tend to show more or less a constant traffic flow throughout the entire day. Based on prior studies, it has been shown that the traffic around any location can be described completely using the Poisson‘s process having inter arrival times described with exponential distribution, we refer to it in our equation as P_{RSU} . Apart from the location and the associated arrival rate of vehicles at the location, another factor which influences the possibility of getting updated from a RSU is the number of such units placed around the country. The number of these units is driven by the QoS requirements of the application and is limited by the investment costs involved in setting up and maintaining these units. We term n as the number of RSUs placed. Using these two terms, we represent the vehicles updated by RSUs as shown in Eq.(6-1).

$$Tr_{toupdate(t)} = nP_{RSU(t)}Tr_{active(t)} \tag{6-1}$$

Considering the fact the number of RSUs are far less, the majority of exchange and spread of information would be relied on the vehicles themselves. Hence, it is important to consider the contributions of vehicles too for spread of updates. Similar to RSUs, the ability to be updated from these vehicles will depend on the number of active vehicles already updated $Tr_{updated(t)}$ as well as on the ability of each of the vehicles communicating with other vehicles. The rate at which vehicles come in proximity of a car can be best described as a Poisson‘s process with inter arrival times described with exponential distribution termed as $P_{V(t)}$. Based on this, vehicles updated by other vehicles at any time t is given by Eq.(6-2). As both of these sources can update other vehicles independent of each other, the resulting vehicles updated from both is summation of the two and is as given below in Eq.(6-3).

$$Tr_{toupdate(t)} = Tr_{updated(t)}P_{V(t)}Tr_{active(t)} \tag{6-2}$$

$$Tr_{toupdate(t)} = Tr_{active(t)}\left\{nP_{RSU(t)} + Tr_{updated(t)}P_{V(t)}\right\} \tag{6-3}$$

One can clearly see from Eq.(6-3) that the vehicles updated at time t, $Tr_{toupdate(t)}$ will be part of vehicles that will act as a source of update at the next time interval $t+1$ $Tr_{toupdate(t+1)}$. Similarly, all the vehicles updated at time instances before time t would act as source. These vehicles updated at previous time might turn off as and when they reach their destination making them unable to further contribute towards spread of information. We represent the fraction of vehicles getting switched off at any time t as α_t. Total number of vehicles is finite in number. As the number of updated vehicles increases, the number of vehicles yet to be updated decreases. The number of vehicles in traffic are finite in nature thus, as number of vehicles getting updated increase the remaining number of vehicles which can actually be updated reduces. This reduction is represented as β_t where $\beta_t \leq 1$. Substituting these changes in Eq.(6-3) we get,

$$\begin{aligned} Tr_{toupdate(t)} = {} & Tr_{active(t)}\beta_t \\ & \left\{nP_{RSU(t)} + \left(\sum_{i=t_{start}}^{t-1} Tr_{toupdate(i)} \prod_{j=t0}^{j=i} \alpha_j\right) P_{V(t)}\right\} \end{aligned} \tag{6-4}$$

Eq.(6-4) ends up revealing the influence of previously updated vehicles on the ability to update new vehicles in the given interval. All these updates that occur at a given time t are due to vehicles updated directly by RSUs or via other vehicles which themselves were updated by RSU or vehicles that were already updated at start of time, ts. We write the equation in terms of these parameters and utilize Eq.(6-4) for $t = t0$. With $Tr_{updated(ts)}$ as the vehicle count with already having the update, we get expression as shown in Eq.(6-5),

$$\begin{aligned} Tr_{toupdate(t0)} = Tr_{active(t0)} \\ \left\{ nP_{RSU(t0)} + Tr_{updated(ts)} P_{V(t0)} \right\} \end{aligned} \tag{6-5}$$

Similarly for time $t1$ we derive the same equation. At time $t1$, we have potentially three possible sources for updates: RSU, the vehicles updated in previous interval $t0$ and from the vehicles which already had the update at the start and are still travelling around. Hence considering these factors in Eq.(6-4) we get equation for vehicles updated at time $t1$ as shown in Eq.(6-6) and rearranging it based on the type of update, we get Eq.(6-7).

$$\begin{aligned} Tr_{toupdate(t1)} = Tr_{active(t1)} \beta_{t1} \\ \{ nP_{RSU(t1)} + \alpha_{t0}\alpha_{t1} Tr_{updated(ts)} P_{V(t1)} + \\ \alpha_{t1} Tr_{active(t0)} nP_{RSU(t0)} P_{V(t1)} + \\ \alpha_{t1} Tr_{active(t0)} Tr_{updated(ts)} P_{V(t0)} P_{V(t1)} \} \end{aligned} \tag{6-6}$$

$$\begin{aligned} Tr_{toupdate(t1)} = Tr_{active(t1)} \beta_{t1} \\ \{ n\{ P_{RSU(t1)} + \alpha_{t1} Tr_{active(t0)} P_{RSU(t0)} P_{V(t1)} \} + \\ \alpha_{t1} Tr_{updated(ts)} P_{V(t1)} \{ \alpha_{t0} + Tr_{active(t0)} \beta_{t0} P_{V(t0)} \} \} \end{aligned} \tag{6-7}$$

On the same lines, we build equations for various time intervals using which we achieve the following generalized expression as given in Eq.(6-8) for spread of updates at any time interval t. This equation reveals Eq.(6-8) has three components that contribute towards the spread of updates: the vehicles updated by the RSUs, vehicles updated by vehicles which were updated by RSUs at previous time periods and vehicles updated by vehicles which were updated by vehicles which already contained the updates at the start of spread.

$$\begin{aligned} Tr_{toupdate(t)} = Tr_{active(t)} \beta_t \\ \{ \\ nP_{RSU(t)} + \\ \alpha_t P_{V(t)} \sum_{x=0}^{x<tn} nP_{RSU(x)} \beta_x . Tr_{active(tx)} \\ \prod_{y=1+x}^{y<tn} (\alpha_y (1 + (\beta_y Tr_{active(x)} P_{V(y)}))) + \\ P_{V(t)} Tr_{updated(ts)} \prod_{y=1}^{y<tn} (\alpha_y (1 + (\beta_y Tr_{active(x)} . P_{V(y)}))) \\ \} \end{aligned} \tag{6-8}$$

6-1-1 Corollary 1: RSUs as Primary Source of Update

A setup where vehicles updated from external mechanism in-spite being a realistic possibility, there will good chances that these vehicles no longer need to update from home if one has a mature map update VANET application in place. Above all, it being an end user action of updating from home/office network there will be days when there are no vehicles which start with updates. For such a setup the equation would be modified as Eq.(6-9).

$$\begin{aligned} Tr_{toupdate(t)} = {} & Tr_{active(t)}\beta_t. \\ & \{ \\ & nP_{RSU(t)}+ \\ & \alpha_t P_{V(t)} \sum_{x=0}^{x<tn} n.P_{RSU(x)}\beta_x Tr_{active(tx)} \\ & \prod_{y=1+x}^{y<tn} (\alpha_y(1+(\beta_y Tr_{active(y)}P_{V(y)}))) \\ & \} \end{aligned} \tag{6-9}$$

6-1-2 Corollary 2: Vehicles as Primary Source of Update

The other extreme being a setup where one does not have any RSUs placed or utilized for spread of updates. Such a setup being ideal in terms of cost and time as one does not need to invest in setting up and maintaining these costly RSUs. For such a setup the equation we get by substituting $P_{rsu(t)} = 0$ in Eq.(6-8) is shown in Eq.(6-10).

$$\begin{aligned} Tr_{toupdate(t)} = {} & Tr_{active(t)}\beta_t \\ & \{ \\ & P_{V(t)}Tr_{updated(ts)} \prod_{y=1}^{y<tn} (\alpha_y(1+(\beta_y Tr_{active(x)}P_{V(y)}))) \\ & \} \end{aligned} \tag{6-10}$$

6-2 Vehicles

The vehicles acting as primary source of update must get map changes from outside VANET from mechanisms other than the use of RSU. Few of the possibilities include use of home/office Wi-Fi network, use of traditional mechanisms such as plugging portable PND to central map server using internet or even use of standard memory devices. Advantages of such a system would be: it is completely a infrastructure free system with no additional expenses required from the map supplier part from providing updates through the standard internet portal. In an infrastructure-less system, the ability of spread of updates relies on the ability of vehicles starting with update to pass on the

Communication Range	350 meters
Data rate	24 Mbits/sec
Map Change	120 Mbits
vehicles	94000
Simulation Time	24Hrs
$Tr_{updated(ts)}$	1-9
Simulation Runs	250 each

Table 6-1: Simulation Setup

information to newer vehicles before their trip ends. In order to obtain maximum possible spread, it is crucial that the vehicles which start the updates and the ones which will get updated by these be of type having maximum travel time and highest possibility to meet other vehicles. This is also reiterated in Eq.(6-10) which shows having $\alpha_y \neq 0$ is crucial to have continuous spread of updates, implying that there is a need for vehicles with update plying around at all times. It also shows that, in order to achieve maximum spread it is crucial to have $\alpha_y \rightarrow 1$ and $P_{V(y)} \rightarrow 1$ implying that vehicles with updates travel the longest and have maximum chance of meeting other vehicles to update. We cannot control the distance a vehicle would travel α_y , nor can we control the ability of the vehicle to meet other vehicles $P_{V(y)}$. Hence referring back at the Eq.(6-10), we can state that predicting the spread of updates is utterly impossible. Instead we decided to observe the spread of updates versus the count of vehicles which act as primary source $Tr_{updated(ts)}$. The ease with which the central server could keep the count of vehicles getting updated using the traditional methods meant that by finding the spread of updates with different number of vehicles we could provide means of approximating the spread of update. It is also crucial to note that vehicles which receive the update by traditional mechanisms do not specify the time at which they would begin their travel and therefore it is crucial that while selecting potential vehicles one must not take time of travel into consideration.

Simulation were executed with simple flooding algorithm as means of dissemination with rest of the parameters as specified in Table 7-5. We randomly picked vehicles of different values of $Tr_{updated(ts)}$ and observed the results. In order to obtain accurate results, we measured the results for different set of vehicles for same $Tr_{updated(ts)}$.

The results of the simulation runs are showcased in Fig. 6-2 reveal the statistical information of average, maximum, minimum and standard deviation for spread of updates. The results reveal the influence of vehicles selected as primary source of update which can be seen in the wide variations of maximum and minimum values of spread of updates. The Fig 6-2 indicates the standard deviation approaching to zero as the number of vehicles as primary source increases indicating that the spread of updates tends to be in and around the average value. It also reveals the influence of a particular vehicle selected for spread of update reduces as the number of these vehicles increases.

We further perform the simulation with larger values covering the entire possible range of count of vehicles so as to be able to generate a valid mathematical equation for the same. The result is shown figure (6-2) and based on it the equation generated is shown in Eq. (6-11). The figure reveals that, within very few vehicles we get considerable number of vehicle updates using using V2V communications. Further, after a critical point any new additions to number of vehicles as primary source, there is no more increase is spread. For our mobility traces, the critical point is 10 vehicles after which no more improvement in spread of updates is observed.

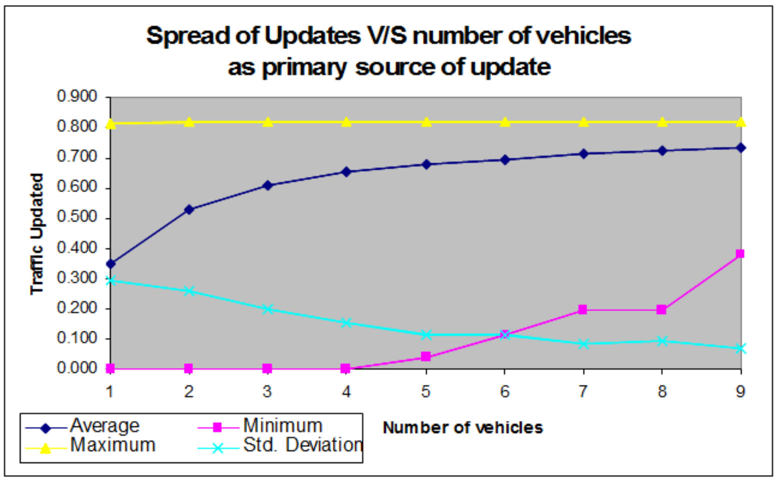


Figure 6-1: Results for vehicle Only Setup

$$\begin{aligned}
Tr_{toupdate(t)} &= (a + b * x)/(1 + c * x + d * x^2) \\
&\text{with} \\
&\text{x} = Tr_{updated(ts)} \\
&\text{a} = \text{-1.1462E+04} \\
&\text{b} = \text{7.8415E+04} \\
&\text{c} = \text{9.9013E-01} \\
&\text{d} = \text{-1.8329E-06}
\end{aligned} \tag{6-11}$$

Our simulation runs revealed that we could achieve 70% of vehicles updated by just 10 vehicles in 24 hours. The results obtained with random selection of vehicles state the ability of system to reach the required spread of updates without any dependency of particular set of vehicles. These results are encouraging for pursuing an infrastructure-less system, but we cannot pursue further with such a system because of the following limitations. Firstly, for simulation we assumed vehicles to be updated using traditional mechanisms to achieve the required spread but this would no longer be practiced by the customers as they would be enticed to get their updates on the fly rather than

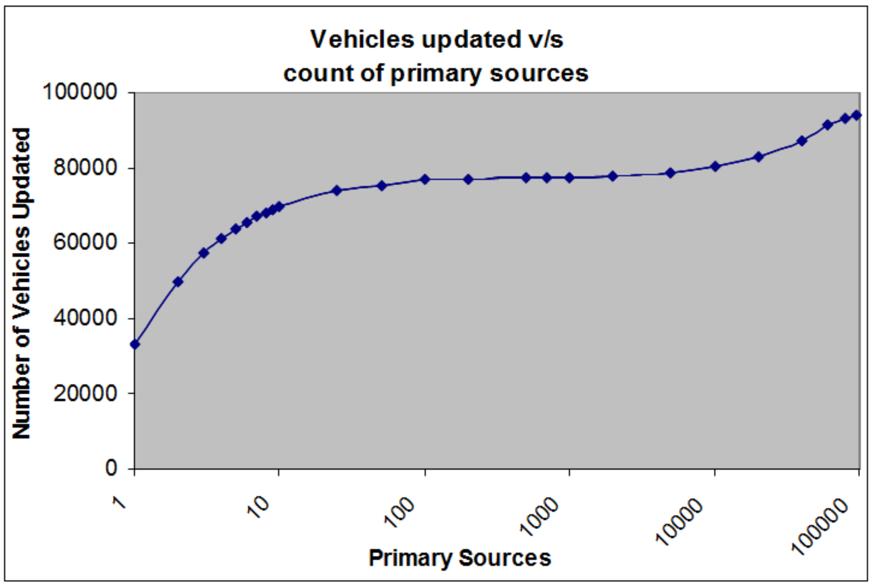


Figure 6-2: Trends for vehicle Only Setup

going through the cumbersome traditional mechanism. Secondly, getting updated by traditional mechanisms does not specify any information on what time they would travel or even whether or not they will travel during the day. Due to these limitations we move onto the other source of update.

6-3 RSUs

Road Side Units (RSUs) are permanent or temporary infrastructure units placed along the roads, and can be used as an alternative source of update. These systems receive the update from central server using backbone internet network (permanent), or based on a portable unit dispatched from the office of the map supplier (temporary) can be used to broadcast the update to the VANET network. One has the freedom to decide on the location of road side units at the deployment stage, it provides a major advantage over the previous setup. The RSUs placed directly, updates the cars which travel in its vicinity. As the location at which the RSUs are placed can be controlled, one can indirectly decide on the mobility parameters α_y and $P_{car(y)}$. Hence, providing a better and consistent spread of updates. This advantage comes at a cost of setting up and maintaining the units. It is crucial to find the optimum setup of road side units to get the best set of results. In this section, we analyze the various configurable parameters associated with it and try to find the optimum solution for the same.

$$\begin{aligned} Tr_{toupdate(tn)} = {} & Tr_{active(t)}.\beta_t. \\ & \{ \\ & n.P_{RSU(t)} + \\ & \alpha_t.P_{V(t)} \sum_{x=0}^{x<tr} n.P_{RSU(x)}.\beta_x.Tr_{active(tx)} \\ & \prod_{y=1+x}^{y<tn} (\alpha_y.(1 + (\beta_y.Tr_{active(x)}.P_{V(y)}))) \\ & \} \end{aligned} \tag{6-12}$$

Compared to the previous setup where there existed only 1 modifiable parameter, for setup with RSUs there are multiple parameters which influence the spread of updates. They are:

- **Number of Units** 'n' : This parameter represents the number of units that would be deployed and used by the application in order to spread updates.

  Higher the number more and more area get covered under the RSUs and lesser dependence on the mobility to spread updates. Higher number will provide more consistent and quicker updates but because of the cost involved in setting up and maintaining the units it is crucial to have an optimum number of units.

- **Location of RSUs** : This parameter is crucial for setups where number of road side units is $n > 1$. Apart from the number of units, the actual location of units which they cover would determine the impact of each of RSUs. The impact is maximum when one has locations such that each of RSUs cover regions such that traffic covered by each of the units is as unique as possible.

- **Duration of Broadcast** 'tr' : This parameter represents the duration of time for which it will broadcast the update. Here, again due to the cost involved with broadcasting updates it is beneficial to have the least possible broadcast duration.

- **Start Time of Broadcast** : The parameter represents the time of day at which the RSUs would initiate broadcast in order to achieve best possible spread.

6-3-1 Location of Units

The emphasis of having least infrastructure support for spreading updates, the responsibility entirely lies with the vehicles to achieve the required spread . Hence it is crucial to have RSUs updating a correct set of vehicles which can spread update to as many vehicles as possible during its travel time. Geographical, economical and social factors influence the traffic and therefore no 2 regions would have the same traffic mobility characteristic same. Therefore placing a RSU at location would not necessarily suffice to provide the best spread of updates. In order to find suitable locations we utilize Eq.

(6-10) and apply limits of number of units $\lim_{n\to 0}$ and duration of broadcast $\lim_{tr\to 0}$. The resulting equation results in a equation shown in Eq.(6-13). The equation reveals that, in order to maximize spread of updates, mobility parameters α_y and $P_{V(y)}$ must be maximum. These are the same parameters that were crucial for updates with vehicle only setup. There we stated that these parameters are not configurable because the vehicles starting as primary sources are of no control for the system. Whereas, in this setup we can decide on the location of RSUs and therefore we can decide which type of vehicles that will get updates from RSU which further will influence the parameters α_y and $P_{V(y)}$.

$$
\begin{aligned}
Tr_{toupdate(tn)} = {} & Tr_{active(t)}.\beta_{tn}. \\
& \{ \\
& \alpha_{tn}.P_{V(tn)}.\beta_{t0}.P_{RSU(t0)}.Tr_{active(t0)} \\
& \prod_{y=1+x}^{y<tn} (\alpha_y.(1 + (\beta_y.Tr_{active(x)}.P_{V(y)}))) \\
& \}
\end{aligned}
\tag{6-13}
$$

The term $P_{V(y)}$ which represents the ability of vehicles to meet other vehicles, will vary for each vehicle for the path it travels and the time at which it travels. In order to find a location which would maximize this parameter, we propose to use locations which have maximum number of vehicles passing through it over the entire duration. These locations give us the best possibility of vehicles interacting with other vehicles and therefore best chances of maximizing $P_{V(y)}$. Further, because mobility is restricted along the roads, the locations in the neighborhood too would provide high probability of communicating with other vehicles.

Use of locations based on number of vehicles passing through it also benefit in maximizing the term α_y . This is due to the fact that by having locations based on number of vehicles passing through it maximizes the number of vehicles which would get the update from RSU. These vehicles in turn would maximize the chances of updating more vehicles and therefore maximizing the chances of achieving $\alpha_y > 0$ for the entire duration of 24 hours.

Procedure

The procedure is a 2 stage process. The first stage finds the count of vehicles which pass through each location. The second stage involves the selection and filtering process. First stage consists of splitting the entire region of Netherlands in grid of squares of each 500 m X 500 m and associating a data structure for each of these grids to maintain a set of cars which passed through it for a given time interval. The procedure for the same is mentioned as Stage 1 in algorithm 6-3-1.

The second stage involves selecting a location with highest count of vehicles followed by removal of it from the list along with all the locations around in the radius of 20 Kms considering the area involved is of 400 Km X 400 Km. We wish to use as minimum possible units. The purpose of performing this is to avoid selecting nearby locations as possible places for RSUs to ensure unique traffic flow between the locations. One can make use of historical data for traffic statistics to find the pattern of traffic flowing between the locations but we decided to make use of distance as criteria of selection as it is far simpler process of selection and at the same time provides equal chances of various regions in Netherlands getting a fair chance of having a road side unit in its proximity. The algorithm for removal is stated in 6-3-1.

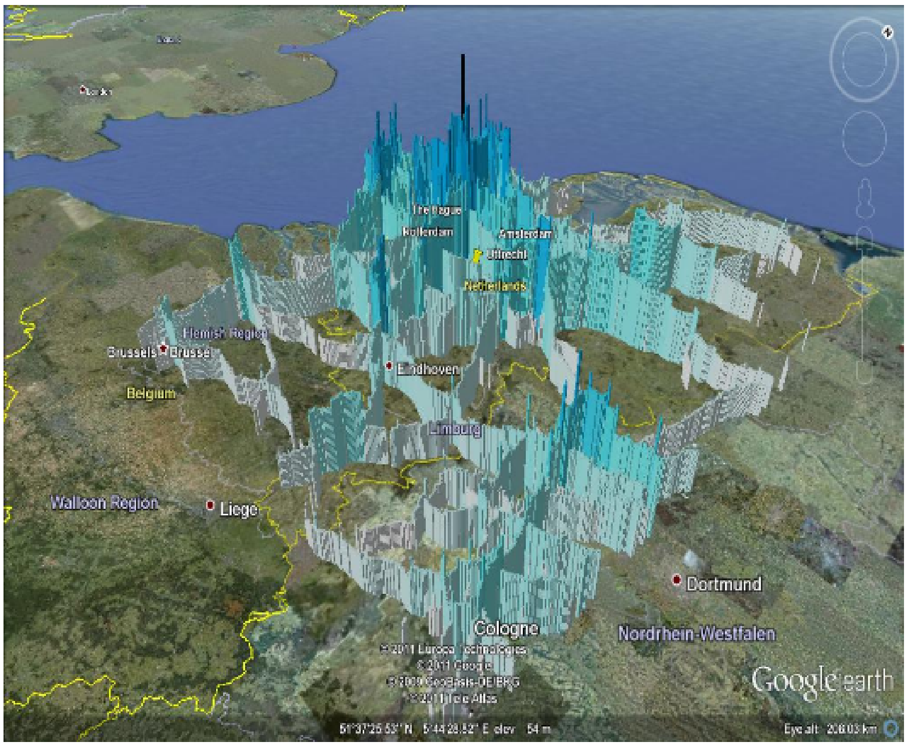


Figure 6-3: Vehicle count V/S Location [46]

Total Traffic common among all 4 locations	0.15
Traffic flow common only to any 1 location	0.10
Traffic flow common only to any 2 location	0.14
Traffic flow common only to any 3 location	0.15

Table 6-2: Top 4 suitable locations

The result of Stage 1 process is depicted in form of a 3D graph 6-3-1 developed using Google Earth with bars placed at locations of the grid block and with the height of

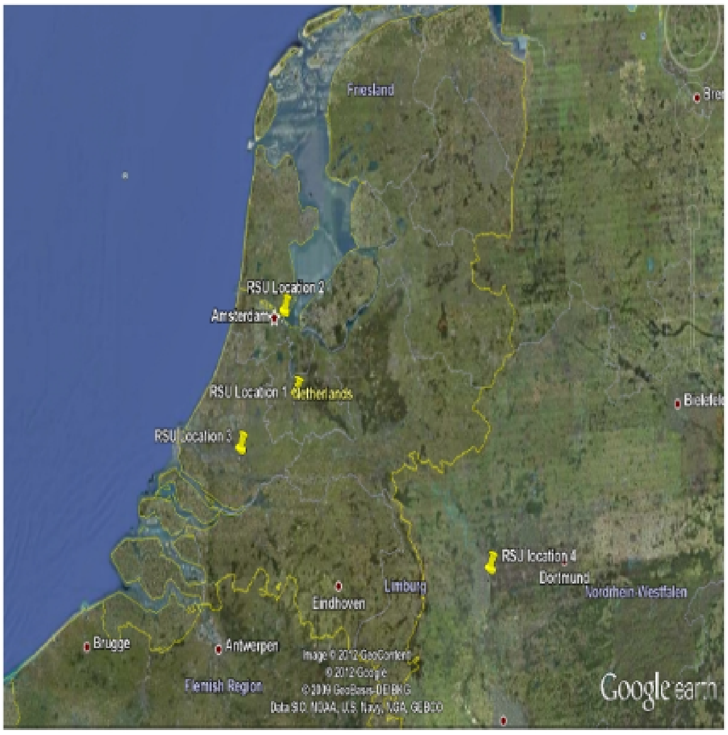


Figure 6-4: **Top 4 suitable locations [45]**

bar representing the number of vehicles passing though the location. The figure 6-3-1 shows the top 4 locations obtained using this strategy and while 6-3-1 indicates the fraction of traffic between any of these locations. What we achieve from this is not only the potential RSUs which have highest possibilities of passing information to other vehicles as well as they are apart so as to have extremely low traffic in common.

Algorithm

Algorithm 5 Find locations for RSUs

```
Require: $N \geq 0, time \geq 0, size = 500m, proximityRange = 20Kms$
  for all $time$ do
    for all $N$ do
      $latId = lat(Car(n))/((latMax - latMin)/size)$
      $lngId = lng(Car(n))/((lngMax - lngMin)/size)$
      $Grid[latId, lngId].count = Grid[latId, lngId].count + 1$
      $grid[latId, lngId] \leftarrow Car(n)$
    end for
  end for                                   {end of Stage 1}
  while $findMaxLatId() \neq -1$ do
    while $findMaxLngId() \neq -1$ do
      $maxLatId = findMaxLatId()$
      $maxLngId = findMaxLngId()$
      $finalRSULocations \leftarrow loc(maxLatId, maxLngId)$
      $disableNearLocations(maxLatId, maxLngId, proximityRange)$
    end while
  end while                                   {end of Stage 2}
```

Algorithm 6 Disable Nearby Locations (disableNearLocations())

```
Require: $N \geq 0, maxLatId \geq 0, maxLngId \geq 0, proximityRange \geq 0$
  for all $Grid[latId, lngId]$ do
    $selLatId = Grid[latId, lngId].latIndex$
    $selLngId = Grid[latId, lngId].lngIndex$
    if $selLatId \leq (maxLatId + proximityRange) \wedge selLatId \geq (maxLatId - proximityRange)$ then
      if $selLngId \leq (maxLngId + proximityRange) \wedge selLngId \geq (maxLngId - proximityRange)$ then
        $Grid[latId, lngId].valid = false$
      end if
    end if
  end for
```

Algorithm 7 FindMaxX (findMaxLatId() and findMaxLngId())

```
Require: $N \geq 0, maxLatId \geq 0, maxLngId \geq 0, proximityRange \geq 0$
  $maxValue = 0$
  $selLatId = 0$
  $selLngId = 0$
  for all $Grid[latId, lngId]$ do
    if $Grid[latId, lngId].countgemaxValue \wedge Grid[latId, lngId].valid = true$ then
      $maxValue = Grid[latId, lngId].value$
      $selLatId = Grid[latId, lngId].latIndex$
      $selLngId = Grid[latId, lngId].lngIndex$
    end if
  end for
```

6-3-2 Number of Road Side Unit (RSU)

A Road Side Unit (RSU) provides connectivity with the internet services and provides a mode of connectivity between disconnected VANETs. The number of RSU is driven by the dependence on ability of a RSU to provide proper functionality of the application. For example: application such as online gaming would always require support of RSU to provide the required connection between vehicles with least latency and maximum throughput. While application like ours, where the support from RSU is very minimal, there is no critical requirement of having a RSU be placed at every road junction. In this subsection we look at influence of number of RSUs on the spread of updates. We analyze the spread of updates, improvements in spread and average delay in updating vehicles w.r.t different number of RSUs. The simulation is executed with simple flooding algorithm for dissemination of map updates and with setup as described in Table 6-3-2.

Communication Range	350 meters
Data rate	24 Mbits/sec
Map Change	120 Mbits
vehicles	94000
Simulation Time	24Hrs
Number of RSU n	1-4

Table 6-3: Simulation Setup

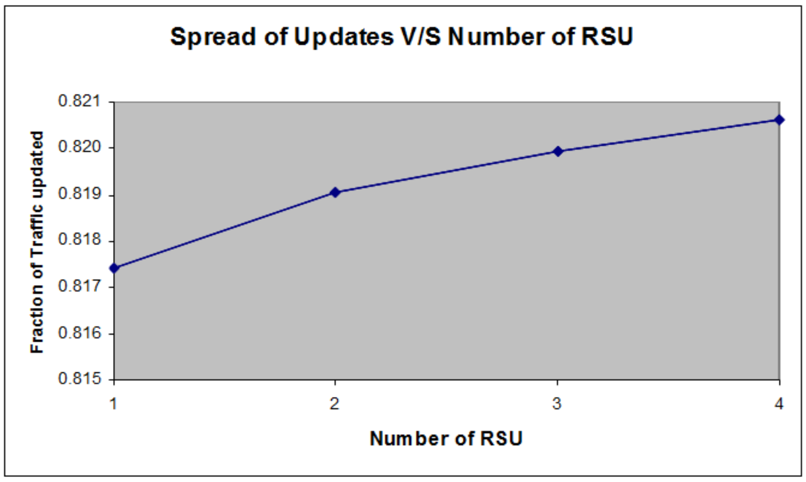


Figure 6-5: Fraction of Traffic Updated for different number of RSUs

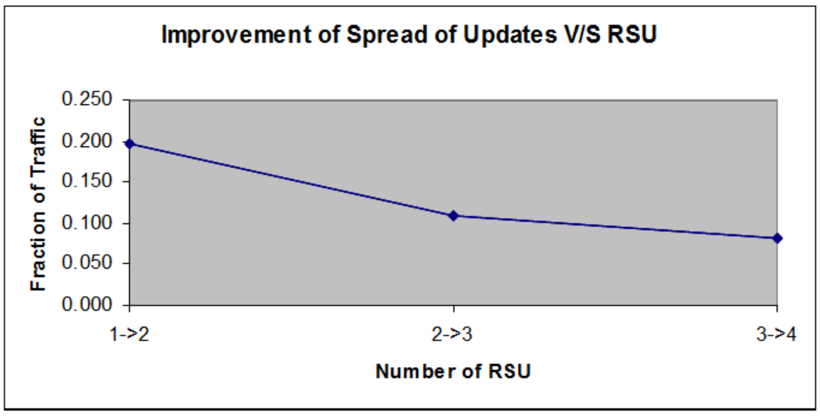


Figure 6-6: Improvement in Spread of Updates with increase in number of RSUs

Spread of Updates

The stats collected are the number of vehicles updated at the end of day and displayed in figure as fraction of overall count of vehicles versus number of RSU. The figure 6-3-2 reveals a gradual increase in the fraction of traffic updated with increase in number of RSU. The increase of RSU to 4, provides an improvement of only 0.38% over the value achieved with a single RSU indicating the improvement to be very minuscule as compared to the investment involved with setting up and maintaining these 4 units. The table 6-3-2 reveals that improvement diminishes with increase in number of RSUs. Table tab:rus has shown that traffic common between the various potential locations is extremely low, drives the point that vehicles updated by each of RSU is different yet the overall achieved spread as shown in Fig. 6-3-2 is minimal. This concludes that, the most of the vehicles updated by each of additional RSUs, interact with each other in due course and would have been updated irrespective of additional RSUs placed. Further, the improvements observed with spread of updates diminish with every addition of RSU.

Overall we can conclude that, the spread of updates obtained froma single RSU is substantial and is approximately 80% of entire set of vehicles. Further, any considerable improvements to spread of updates require substantial increase in RSUs as gains obtained are minimal and diminish with every addition of a RSU.

Rate of Updates

The statistics collected are number of vehicles updated per hour and is displayed in figure as fraction of number of updates that can be achieved with a single RSU. The purpose of plotting the data w.r.t spread of updates with respect to that observed with one RSU is to investigate the improvement in rate of updates one can achieve with

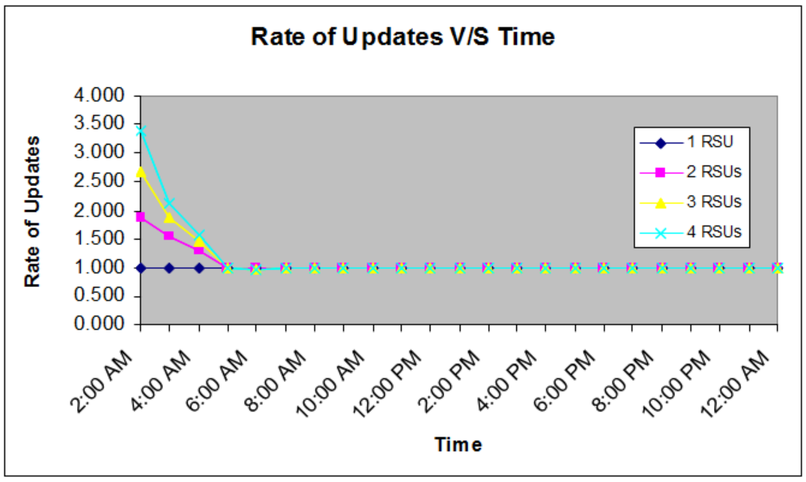


Figure 6-7: Rate of Updates for different number of RSUs

additioal RSUs. The Fig. 6-3-2 reveals that for all the configurations the rate of spread is proportional to the number of RSUs at start $R_{update} \propto n$. But as time progresses the improvements diminish until 4 hours have passed where the number of RSUs do not provide any improvements and rate of updates as same as that can be achieved with a single RSU. Hence we can conclude that the advantages of addition of RSU lasts for only few hours and after which there are no more improvements of adding these units.

Average Time of Update

The stats collected measures the time it took to update vehicles to get updated from the start of their first trip in the day. The Fig. 6-3-2 shows the average time of updates for the traffic updated for different combinations of four RSUs. The figure reveals the average time decreases with increase in number of RSUs. This is mainly accomplished due to the increase the rate of updates seen in the early hours of simulation. The sole advantages of adding RSUs can be said to be of achieving a smaller delay in updating the vehicles

6-3-3 Time of Broadcast

The time at which the Road Side Unit (RSU) starts issuing updates is a paramter of the system that will influence the spread of updates. Ideally, the time of issue of update would like to be such that by the end of the day, maximum set of vehicles are updated. To find how the time of issue of update impacts the overall spread of updates, we simulate the spread of updates for 12 different values of start of time of issue of broadcast separated by 1 hr from each other, starting from start of the day. The results are shown in Fig. 6-3-3, which reveals that as time of issue of updates is

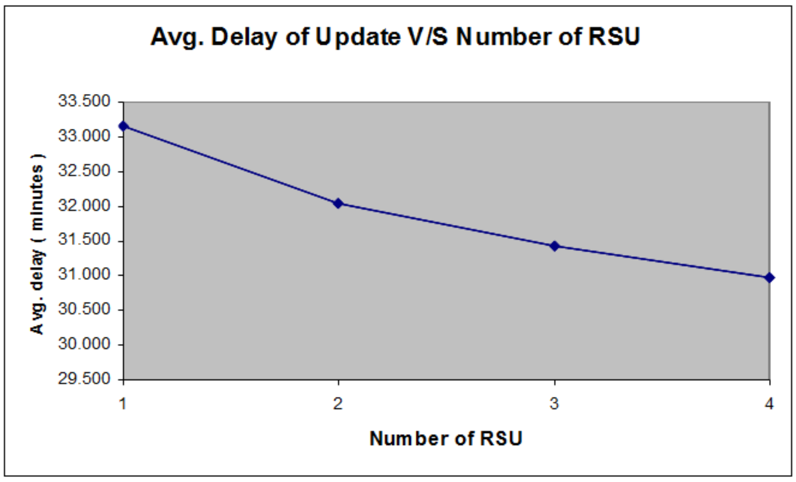


Figure 6-8: Average Time of Updates for different number of RSUs

shifted in the later part of the day lesser fraction of traffic is updated. The reduction in traffic updated remains fairly minimum during the early morning hours but then rapidly decreases during the time interval which coincides with the increase in traffic around 6AM. This concludes that the best possible time for which to issue update are early hours of the day before which traffic picks up for morning office hours, which enables the system to have enough vehicles with update such that they are able to spread updates to maximum set of vehicles during the morning rush hours.

6-3-4 Time Period of Broadcast

Duration for which the Road Side Unit broadcasts the update is another crucial parameter that will influence the spread of updates. This parameter is represented as tr in Eq.(6-12) and once can see that higher the value higher the spread of updates one can achieve. To observe the impact we simulated the spread of updates for 5 different time intervals (1, 6, 12, 18, 24). Fig. 6-3-4 describes the updates of vehicles directly from RSU while Fig. 6-3-4 shows that different time interval has hardly any impact on the spread of updates. This is due to the fact that once the RSU updates vehicles passing by, they in turn will update other vehicles approaching the RSU. As the time passes by, more and more vehicles will be updated by the vehicles itself and hardly rely on RSU to support further spread of updates. Further, we can state that the parameters $P_{RSU(t)}$ and $P_{V(t)}$ decrease over time for which RSU is deployed or the vehicle is running with update. The reduction in values of $P_{RSU(t)}$ and $P_{V(t)}$ compensates any increases in values of tr has on the spread of updates.

Crucial factor that would influence the decision of time of broadcast would be the market penetration. Market penetration is defined the as the fraction of entire set of vehicles which would have adopted the IEEE 802.11p technology by installing the

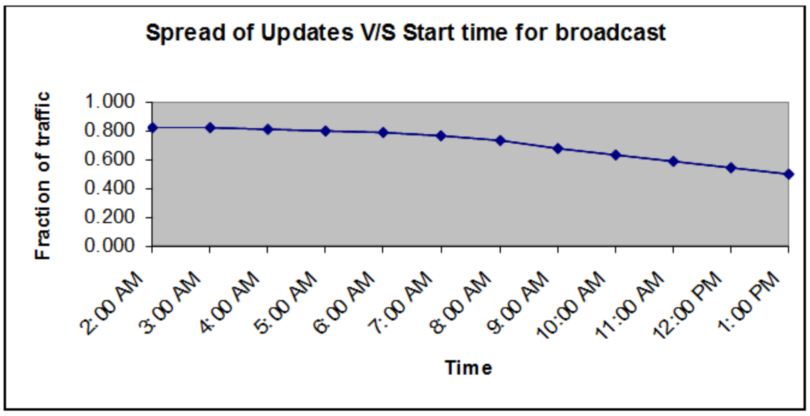


Figure 6-9: Time of Broadcast w.r.t Time of Issue of Update

required hardware. Market penetration decides the ability of vehicles to be updated by RSU $P_{RSU(t)}$ as well as the ability of updating from other vehicles $P_{V(t)}$. Therefore to observe the influence of period of broadcast on the spread of updates for different market penetration, we generated results for different market penetrations (10, 30, 50, 70 and 90%) for 2 different durations of broadcast (6hrs and 18 hrs). The results are shown in form of a Fig. 6-3-4 where improvement of spread of updates is plotted against different market penetrations.

The Fig. 6-3-4 reveals that improvements are nearly 4 times when the market penetration is low as 10% and the improvements decrease as market penetration increases with near 0% improvements seen when market penetration increases above 70%. This concludes that for lower market penetration it is very vital to have longer duration for broadcast so as to achieve best possible spread of updates.

6-4 Summary

This chapter provided details on the type of sources and the influence of its various parameters on the spread of updates. We began our discussion with an Infrastructure less system where vehicles themselves act as the primary source and update other vehicles in VANET. This is possible with vehicles getting the update from Home or Office WiFi network rather than on dedicated RSU to provide it with the update. Simulation results showed that such a system is sensitive to the selection of vehicles which act as the primary source with results varying over large extremities. Results also showcased that this issue can be mitigated by the having more and more vehicles starting with update. They also showed that the results obtained are quite close to that one can achieve using RSU.

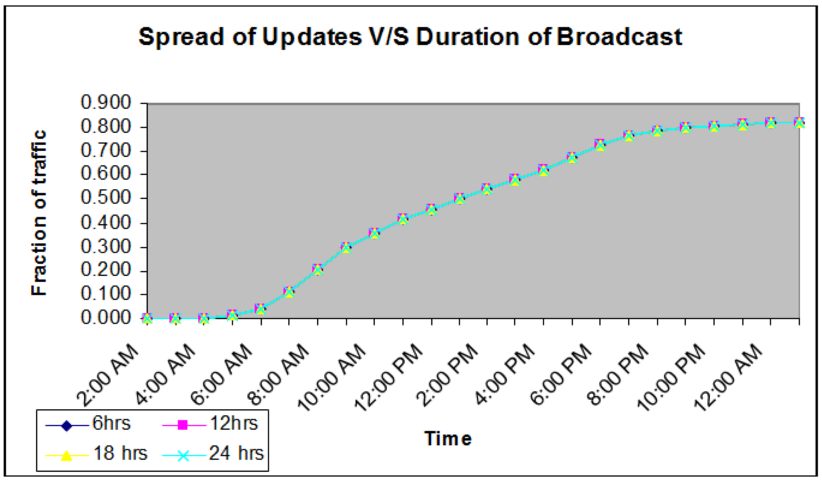


Figure 6-10: Spread of updates V/S Duration of Broadcast

The inability of able to control the number of vehicles and the time at which vehicles actually start the travel with update makes an Infrastructure less system not practical to implement. Therefore we shift to an Infrastructure based system where updates are provided by RSUs. First we discussed the strategy of selecting locations of RSU which involves using mobility traces which are most likely to come across different vehicles. Using this strategy we could come up with 4 distinctive locations, all on the major junctions of highways in and around Netherlands.

The study of influence of number of RSU showed that the improvements obtained by increasing the number is extremely low and decreases further with every increase in number of units used. Study also showed that the addition of RSUs improved the average time for vehicles to get updated. It also revealed that improvement of spread of updates obtained by addition of vehicles is only for the initial 4 hours of update and performance fairly remains the same for remaining part of the day. We further investigated on influence of time of broadcast and duration of broadcast has on the spread of updates. The results revealed time of broadcast has least influence on spread of updates for early hours of the day, but if we delay the time of broadcast spread of updates is heavily affected and is unable to obtain the best possible spread of updates. Further, the duration of broadcast has no influence on spread of updates with 100% market penetration but for lower market penetration, the duration of broadcast plays vital role in deciding the effective spread of updates reached.

Having seen the influence of various types of sources and parameters we can conclude this chapter with saying that RSU as source of update benefits from Infrastructure less systems as one can predict the approximate spread of updates. Further, the selection of location if done based on the intensity of traffic at the location one can achieve more than 80% spread with just one RSU broadcasting for first one hour of the day. We would consider this as our setup for next chapter where we discuss the dissemination

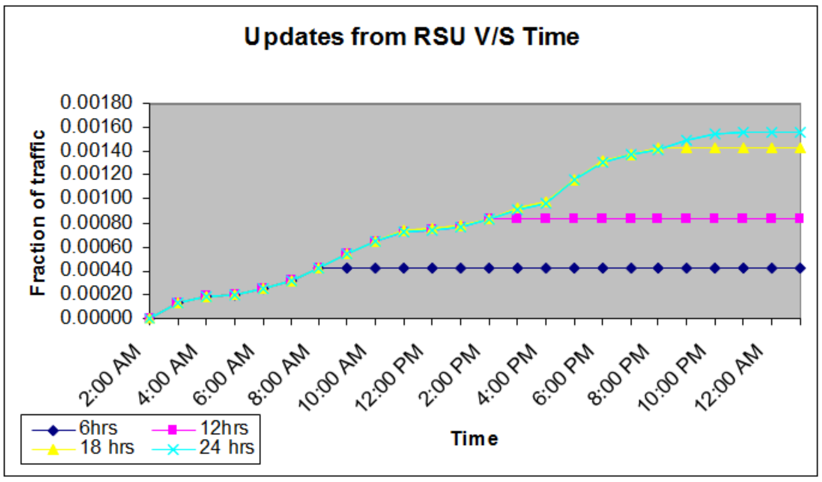


Figure 6-11: Spread of Updates to vehicles from RSU V/S Duration of Broadcast

algorithm for achieving the spread of updates.

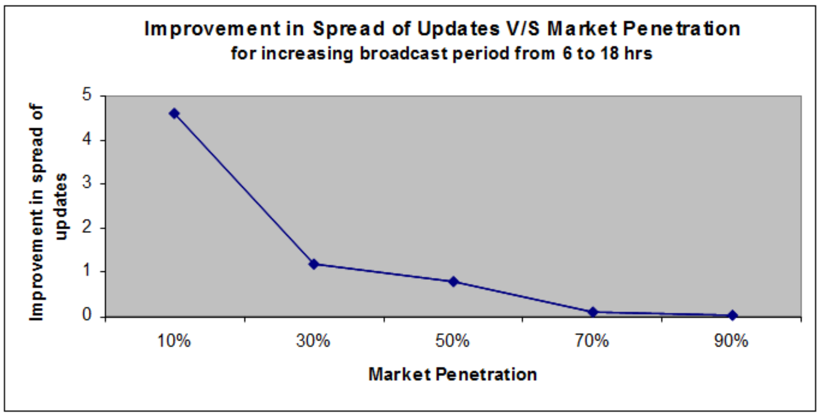


Figure 6-12: Time period of Broadcast w.r.t Time of Issue of Update with Market Penetration

Chapter 7

Data Dissemination Algorithm

7-1 Introduction

We discussed mobility characteristics and influences of various system parameters on the spread of updates and the implementation of simulation tool in earlier chapters. Now we move to another vital aspect associated with our application. We discuss the suitable dissemination algorithm to be used for our application which can achieve the best possible spread of updates with least overhead and efficient use of resources. We discussed the previous chapter with simple flooding as our dissemination algorithm and hence we start our discussions with its advantages and limitations. Then we discuss eneric methodologies: top level and bottom-up approaches with which dissemination algorithms are built. We follow it up with discussion on algorithms based on top-down approach and show their simulation results. Further we show the limitations with this approach in a dynamic system like ours. To resolve this, we propose a dissemination algorithm based on bottom-up approach. In this we propose two flavors and compare the performances of them with respect to previously discussed strategies.

7-2 Flooding

In the previous chapter, we discussed the influence of various configuration parameters on the spread of updates with using flooding as the dissemination algorithm. Strategy in flooding involves vehicles, which when updated keep broadcasting the data until they exit from VANET. Results, as seen in the previous chapter, indicate the flooding algorithm to be successful as it can achieve updates of around 80% of all vehicles in 24 hours. The algorithm is able to do so because of the following facts: (i)the algorithm is simple and need not control data to be passed along with actual data. This makes the algorithm utilize its entire link availability time on actual data exchange. This avoids the wastage of bandwidth on control messages, thus making use of connectivity among vehicles. (ii)it does not require gathering of information about its neighbors to make a decision on dissemination, which avoids overhead of exchanging messages and saves the precious link availability time. Apart from its dissemination capabilities, it also provides resilience to link loss, by providing consistent dissemination with heavy losses. Fig. 7-2 shows that even with a link loss probability of 0.7, the system is able to reach 70% of vehicles, with only loss of 20% in performance. Further, because it does not make any assumptions of the system or the environment, it is favored for deployment in any traffic scenarios without requiring any changes.

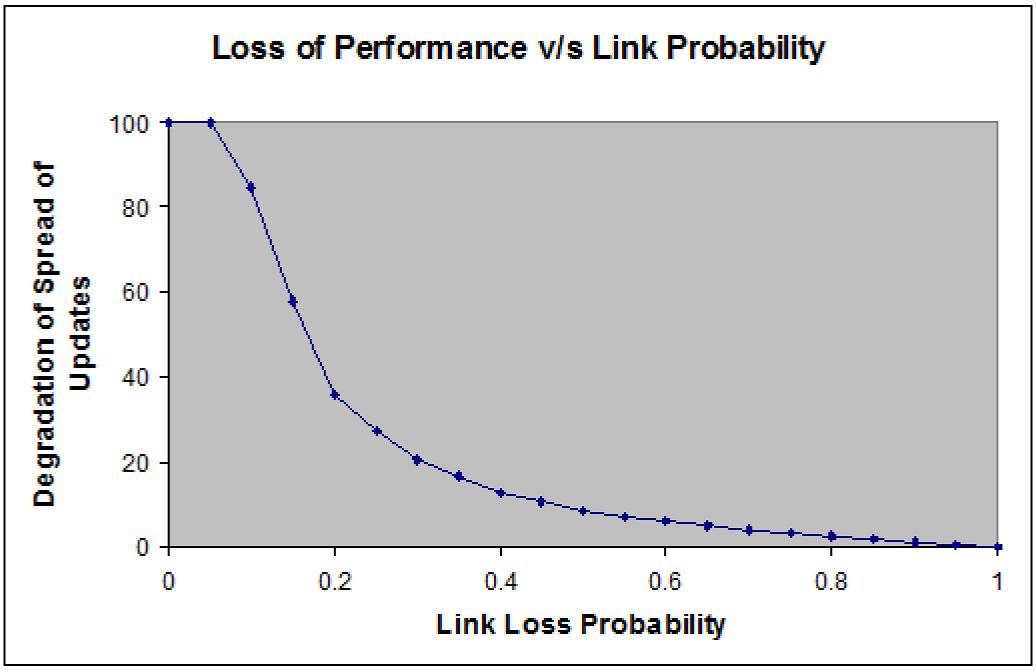


Figure 7-1: Loss of Performance of Spread of Updates V/S Link Probability

In spite of all the advantages mentioned, flooding is rarely used in its standard form due to the disadvantages associated with it. The primary problem is the issue of broadcast storm. As more and more vehicles would get updated, they would start themselves to act as broadcasters. This results in rapid increase in the count of broadcasts increasing contention for channel for broadcast, and thus collisions and redundant messages. Further, lack of information about the status of one‘s neighbors results in having unnecessary updates among the vehicles even when all the neighboring vehicles have already been updated. Furthermore, having no selection criteria for next potential broadcaster results in situations where redundant broadcasts being done though all the vehicles are already updated.

A simple setup with one RSU with flooding strategy is used to calculate the extent of number of redundant messages and collisions occurring over 24 hours. The results obtained from it are displayed in Table 7-2, which reveal that the number of redundant messages and collisions are in the order of 42 billion and 6 billion respectively! Such large numbers indicate a huge loss of communication time and communication channel as a resource, making the strategy inefficient and hence being unsuitable to be used in

Parameter	Value
Number of Redundant Messages	42046841
Number of Collisions	6875969

Table 7-1: Overall Count of Redundant and Collisions with Flooding Mechanism

our application. Before we discuss our algorithm, we would like to discuss the reasoning behind selecting a broadcasting algorithm over a unicast based one. Unicast algorithms, involves point to point communication between the nodes, which is executed by use of identifiers which uniquely identify every node in the network. These mechanisms are used in situations where one needs to communicate with a single or a small subset of nodes from the entire set. This type of strategies help to avoid redundant messages as one performs handshake between the parties to indicate whether they wish to participate in communication or not. Further, the vehicles in communication would have information about the path the packet has to travel helping the packet to reach the end destination in the fastest possible way. In spite of all these advantages we select the broadcast based algorithm because:

- **Intended Recipients** : In our application , every vehicle fitted with TomTom‘s navigation hardware would require the update and considering that TomTom has a large market share in navigation systems, we can very well state that large fraction of entire set of vehicles would be in need of the update. Using unicast algorithms becomes an overhead because the transmitting vehicle would require all the vehicles to provide it with identifiers so that it can then issue the update to each of them. Further, having individually updating each neighboring vehicle would imply that the n^{th} vehicle would have to wait for n time intervals to actually receive the update, whereas in broadcast all the vehicles would be updated at the same time. This concludes that use of unicast algorithm would require n times the time required by a broadcast based algorithm to update n neighboring vehicles. As we have seen in Chapter 4, ability of vehicles to remain in proximity of anther vehicles decreases rapidly with increase in time. This implies that the spread achieved by unicast algorithms would be far less as compared to that achieved by broadcast algorithms.
- **Mobility Issues** : Mobility, an integral part of VANET adds to the overhead in unicast algorithms as it needs to keep track of its neighboring vehicles. This is supported by empirical analysis conducted in Chapter 4, where we showed that traffic in VANET is continuously changing. The requirement of keeping track of neighbors reduces the time available to actually spread the updates among the neighbors and therefore reduces the spread of updates achieved by unicast algorithms.
- **Inherent advantages of broadcast**: In most scenarios, the same update is required by each and every neighbor. Hence omni-directionality of broadcast of reaching all vehicles at the same instant is lost in unicast algorithms resulting in poorer performance compared to broadcast based algorithms.

7-3 Design Considerations

Using the empirical analysis conducted in Chapter 4 we list a few requirements that must be taken into consideration before designing the algorithm. They are,

- **Spread of Updates**: The primary objective is to achieve the best possible spread of updates among the entire set of vehicles over entire 24 hours.
- **Efficient Algorithm**: The spread of algorithm must be achieved with least possible contention for channels, least redundancy and collision of update messages.
- **Duration of broadcast**: Analysis revealed that vehicles get activated over different time intervals over the day, with hardly $\frac{1}{4}^{th}$ of entire set of vehicles being active at any moment of the day. This makes it necessary for the algorithm to be broadcasting for over the entire 24 hours of the day.
- **Support wide range of traffic levels**: Analysis also revealed that traffic varied over wide ranges in the entire 24 hours. Early and late hours have very low traffic count whereas day time showcases to have consistently high levels of traffic. Time around opening hours and closing hours of work particularly show rapid changes to the traffic. All these characteristics of mobility imply that our algorithm must be able to work in the every traffic scenarios as well as adapt well to rapid changes to traffic.
- **Neighbor vehicles**: Analysis revealed that number of vehicles in proximity vary for every vehicle as well as the instances of it occurring also vary over wide ranges. All these imply that the algorithm need not assume any characteristics about the neighborhood. Due to these wide variations, the algorithm must avoid having to exchange information with one‘s neighbors.

All the above requirements can be satisfied with flooding technique except for the requirement of efficient algorithm. Hence, we decided to develop an algorithm which can take advantages of flooding but overcome the issues of efficiency associated with it. Flooding algorithm has been already studied in different networking domains and various solutions have been proposed. These solutions have mainly been based on methodology of top-down approach. In this approach, characteristics of the system are determined by parameters at the global level. Few of the examples include use of limited number of broadcasts or a time limit after which the broadcasts are stopped. Based on this approach we propose following strategies

- **Fixed Duty Cycle**: Broadcast in flooding algorithm can be considered as a signal with 100% duty cycle where the broadcasts occur at every possible time. This requirement if relaxed will directly reduce redundancy and collisions associated with flooding algorithm. Further, the mobility of vehicles is restricted on roads and has limitations such as, maximum speed of a vehicle itself. This implies that varying the duty cycle should not degrade the performance of spread of updates to be unacceptable. Hence to observe the effects of duty cycle we propose five different duty cycles (10, 30, 50, 70 and 90%) and observe the responses of spread of updates along with improvements in redundancy and collisions. The Table 7-

Duty Cycle	Spread	Redundant messages	Collision messages
10%	48838	3432028	741879
30%	69213	11979236	2145987
50%	72712	19942031	3424389
70%	74816	28037467	4690432
90%	76164	36147813	5960637

Table 7-2: Spread of updates for different duty cycle of broadcasts

3 shows that spread of updates fairly remains constant for variations in duty cycle. With duty cycle of 50% variations provide hardly a drop of 5% of spread of updates but provide more than 50% reduction in number of redundant messages and collisions as compared to that seen in flooding algorithm.

- **Linear decreasing duty cycle**: Previous strategy involved considering a constant value of duty cycle for the entire 24 hours. The mobility characteristics and the pattern of updates showcase that during early hours the number of vehicles are less and so are the number of potential broadcasting vehicles. While as time progresses the number of vehicle with broadcast increase too. Thus vehicles can co-operate to issue updates. Hence we propose the use of duty cycle with characteristics of a constant negative slope. We tested it with different slopes and the results are shown in Table and the graphical representation of duty cycle is shown in Fig 7-3. Note that the negative slopes indicate an reduction in duty cycle.

Slope	Spread	Redundant messages	Collision messages
-1	74132	27026743	4473114
-(2/3)	75368	31668906	5225509
-(2/4)	75540	32410730	5348178
-(2/5)	75435	31460082	5204178
-(2/6)	75719	33206417	5476101
-(2/7)	75879	34281855	5658446

Table 7-3: Spread of updates for linear decrease of duty cycle of broadcasts

Table 7-3 shows that, more aggressive the decrease of duty cycle more improvements are observed in redundant messages and collisions. At the same time, it is also accompanied by poorer spread of updates.

- **Non linear decrease in duty cycle**: A non linear change of duty cycle for broadcasts may compensate for the non-linear characteristics of updates. Hence to test the performance, we simulated the spread of updates with characteristics

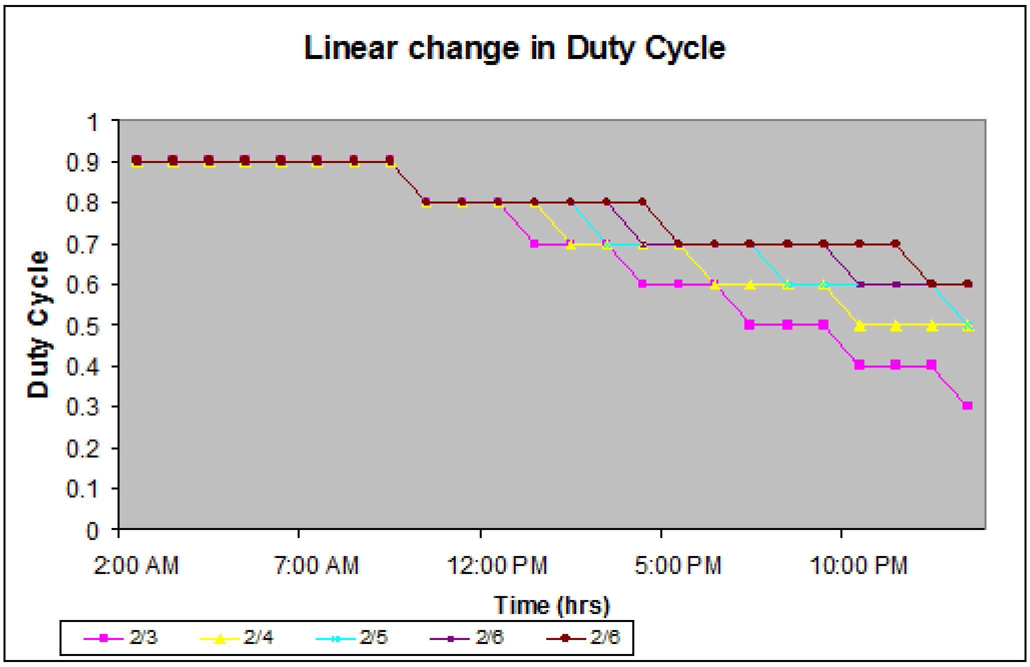


Figure 7-2: Constant decrease of duty cycle

of $Cycle_{new} = Cycle_{curr}/m$ where m is considered having $\frac{3}{2}$, $\frac{4}{2}$, $\frac{5}{2}$ and $\frac{7}{2}$ as value. The graphical representation of duty cycle is shown in Fig ??.

Duty Cycle	Spread	Redundant messages	Collision messages
(3/2)	73093	25267417	4185247
(4/2)	69824	21443461	3559551
(5/2)	64357	19252553	3191784
(6/2),	64183	18748586	3110319

Table 7-4: Spread of updates for non-linear decrease of duty cycle of broadcasts

Table 7-3 reveals that drop in performance of spread of updates gets higher as the reduction of duty cycle becomes more aggressive. For example with $m = 4$ the degradation is of more than 9% with number of redundant and collision messages being only 55% and 54% of that observed in flooding algorithm.

- **Time slot**: All the above strategies discussed where we vary duty cycle provided some improvements to redundancy and collisions at the cost of slight reductions in spread of updates. To further improve, we propose to assign different groups to broadcasters. The groups are assigned in a way such that every group has a duty cycle that is phase shifted as compared to previous. The phase shift is calculated as $Phase = (360/N_{groups})$. This is help to reduce the number of contentions among the vehicles as well as provide a far more dispersed updating opportunities. We tested this approach with N_{groups} of 2,4,6 and 8 with results as shown in Table 7-3

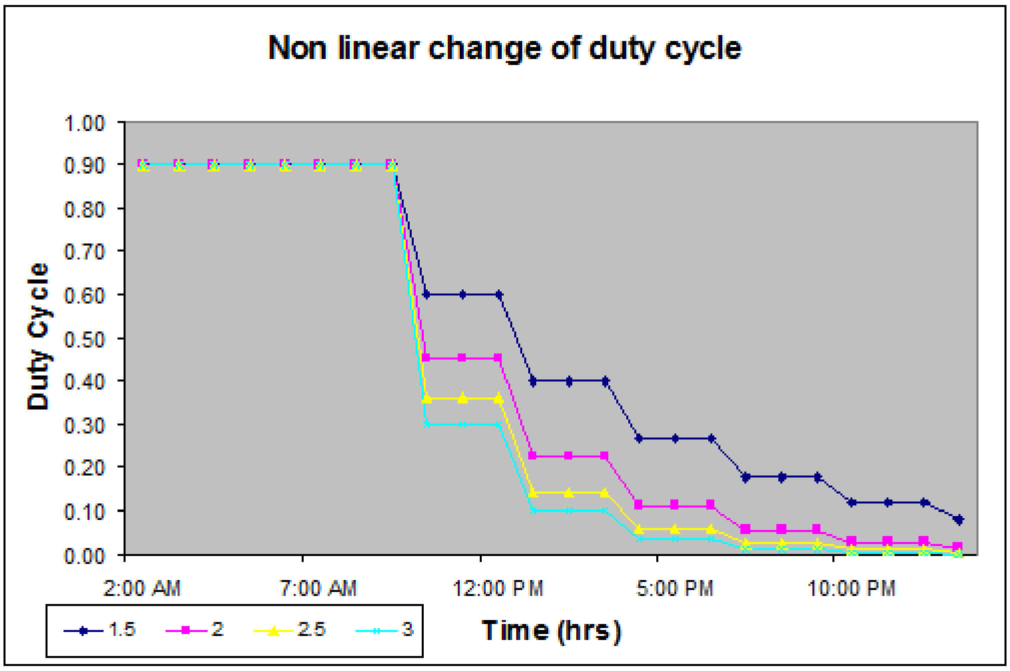


Figure 7-3: Non linear decrease of duty cycle

Duty Cycle	Spread	Redundant messages	Collision messages
2	74002	23870344	3993650
4	71031	16385681	2780069
6	68386	12391131	2565467
8	65475	10261820	2437431

Table 7-5: Spread of updates for time slot based system

Table 7-3 reveals that drop in performance of spread of updates gets higher as the number of groups increase. For example with $N_{groups} = 8$ the degradation is of more than 13% with number of redundant and collision messages being only 75% and 64% of that observed in flooding algorithm.

All the above discussed strategies showed improvements but with a certain loss in performance in spread of updates. To understand and compare the algorithms we plot two graphs, redundant message against number of vehicles updated Fig 7-3 and collisions against number of vehicles updated Fig. 7-3 .

Fig 7-3 reveals that all of the duty cycle based algorithms tend to show a similar pattern of reduction of redundancy with the simple constant duty cycle based algorithm to be the best of the lot as it is capable to provide the best spread of updates amongst other duty cycle based algorithms. This also holds true in case of collisions which can be seen from Fig 7-3.

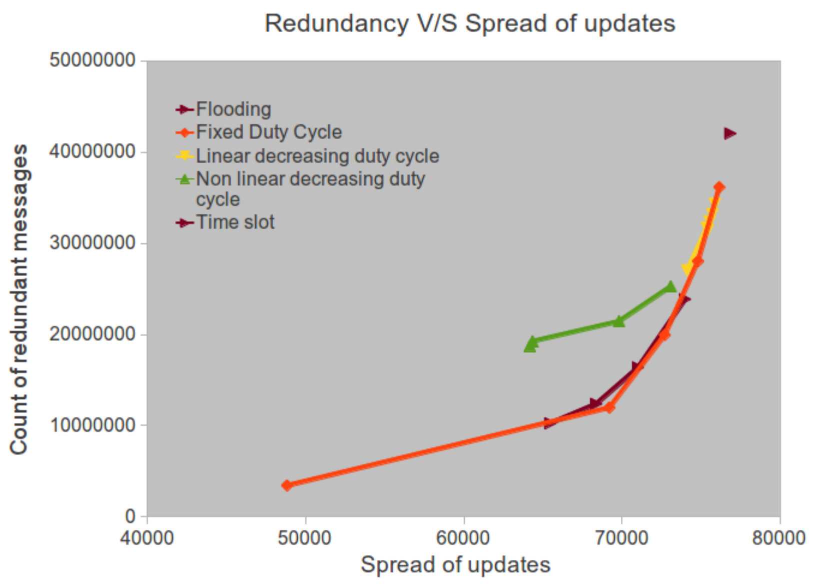


Figure 7-4: Reductions in redundancy for different system level strategies

The comparisons of algorithms in Fig 7-3 and Fig 7-3 reveals light into limitations of top level design which is seen for each and every algorithm discussed. For example, Duty cycle variations provide reductions to redundancy and collisions because we control the number of broadcasts which indirectly reduces redundancy and collisions. But this comes at the cost of reduction in spread of updates. If one considers two different locations, a city road with multiple updated vehicles and a less travelled highway with only one updated vehicle, an algorithm which uses a global value for parameters to make its decision, cannot provide best results for each of the locations. Having higher duty cycle would provide best spread at the less travelled highway but would also increase the chances of redundant messages and collisions at the city road, while a lower value of duty cycle would provide the mechanism to control the redundant messages and collisions but would not be able to provide the best spread of updates at less travelled highway. Whatever values we select one location benefits at the cost of other. The dynamic characteristics of VANET consisting of continuously changing network topologies, there is a need for decision making to be done at local level rather than at a global level. Further, there is also a need for having a sort of feedback with which each vehicle can update its parameters based on its local network characteristics. This approach is known as the bottom up approach where the decisions are made at the local level and system wide characteristics emerge at global level. We try to meet the requirements with our algorithm Adaptive and Distributive Broadcasting algorithm (ADB).

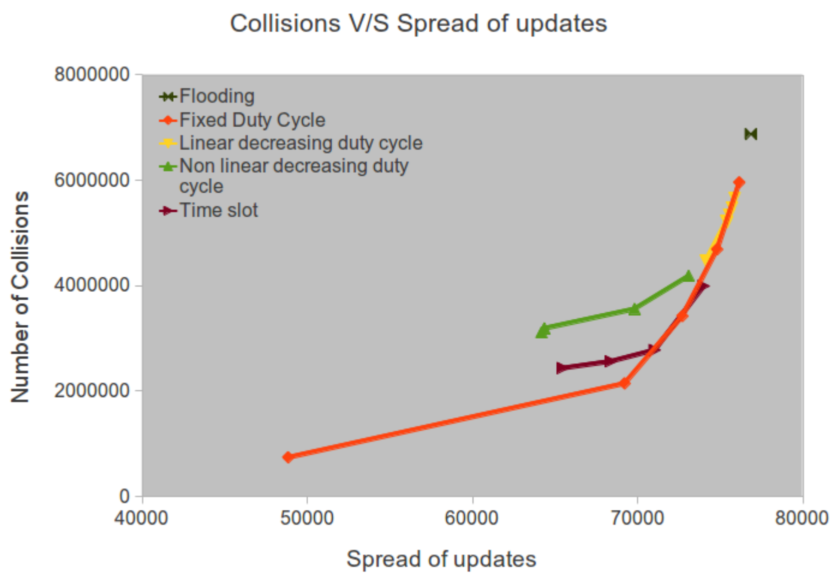


Figure 7-5: Reductions in Collisions for different system level approach strategies

7-4 Adaptive and Distributive Broadcasting algorithm (ADB)

Adaptive and Distributive Broadcasting algorithm (ADB), is a dissemination algorithm proposed to achieve a distributive and co-operative mechanism for broadcasting data among all the vehicles in VANETs. The decisions for broadcasting are primarily based on simple but effective mobility concepts with an objective to achieve maximum possible spread with least overhead and low count of redundant and collision of messages. Further, it is designed without any assumptions about mobility traces used or the environment such that it can be used without any modifications in any traffic scenarios in any part of the world.

7-4-1 Assumptions

The algorithm has been designed based on few assumptions like each and every vehicle part of the dissemination is fitted with GPS and has a TomTom provided digital maps. Each of these vehicles has been fitted with the same hardware such that each one has the same communication capabilities and the same piece of software running. Further each of them has the knowledge of their destination of travel and the approximate time of travel and is available to the communication software.

7-4-2 Concept

Spread

Flooding algorithm showed that dissemination is best achieved with an aggressive approach towards broadcasting. Therefore we base our algorithm on aggressive approach but with mechanism to control it so as to avoid the problems of flooding algorithm. Duty cycle based approach showed that one could regulate the duty cycle and yet manage spread of acceptable value with lower redundant message and collisions. Hence we use duty cycle to control the aggregation and regulate it by varying the duty cycle. The regulation is attained by use of pseudo-feedback mechanisms, which is used to gather information about presence or absence of broadcasting vehicles in the vicinity. The information of vicinity is gathered using excess or lack of redundant messages and not on request-reply mechanism and hence the name pseudo-feedback mechanism. These mechanisms are explained in the following subsections.

Redundancy and Collisions

Redundancy occurs when the vehicle updates its neighboring vehicles even though it has already received the update but the broadcaster has no information about it. One of the techniques is to ask for status update from its neighboring vehicles and then make a decision on whether or not to issue the update. This can be accomplished by exchanging map version which each vehicle is using and based on which the broadcaster can decide on whether or not to broadcast the data further. This technique will work but would require one request, n replies from n neighbors which would require in best case scenario $n + 1$ times to exchange data without any collision of data. Further, this time compounds when more than one vehicle wishes to obtain status information from vehicles in the same vicinity. This makes the solution not operable in the practical world of VANETs.

We propose a new technique where rather than relying on explicitly asking for map version, we listen to the update broadcasts in one‘s vicinity. This way we know information about a vehicle which is in our vicinity and also has the update. In such situation, we just avoid trying to broadcast. This not only would reduce number of broadcasts but also avoid redundant messages and hence avoid collisions too.

This technique is also applied for vehicles which have just received the update. These vehicles need not start broadcasting immediately as the vehicle which updated this vehicle will be in proximity for certain time duration and can suffice the needs of nearby vehicle that is yet to be updated. Further, this is extended during the entire lifecycle wherein, as soon as a potential broadcaster receives a redundant update it can prolong its inactivity or stop issuing broadcasts for time duration until these vehicles are no longer in proximity.

Inactivity

The effectiveness techniques specified for reducing redundancy and collision lies on the ability of the algorithm to calculate the time duration for which it needs to remain inactive such that no longer vehicles with update are in proximity. This is crucial because, if the value of time is kept high it would involve possibility of having the non-updated vehicles coming in proximity to these vehicles and yet do not get updated whereas on the other hand if kept too low would result in contention for channel and therefore increases the redundant updates and collisions. So to calculate accurate information one of the techniques is to exchange details such as destinations of vehicles, directions with its neighboring vehicles. This information can then be used to easily calculate the accurate time required for vehicles to be far off from each other. But this exchange of private information results in loss of privacy and causes security concerns during implementation of the algorithm. Hence we propose an alternative solution where we propose to use lack of broadcasts in its vicinity to interpolate the same. This way we not only accurately get whether the vehicle with update is farther away from its communicating range but also find whether any new vehicles have been updated are in the vicinity.

Balance of number of broadcasts

The techniques specified above contain decision making for the receivers and nothing for the broadcasters. These broadcasting vehicles would still end up broadcasting the redundant update until someone else broadcasts before it does. Hence, this would add up to the redundant and collision of messages. Further, the techniques do not specify how to handle broadcasts when there are no vehicles around. To resolve it, we propose a strategy where, after every broadcast, it monitors the time duration for which it failed to receive any update from any neighbor. When this count crosses a threshold, it can stop broadcasting update.

7-4-3 Communication Lifecycle of a Vehicle

We propose a set of communication states in which a vehicle would operate during its entire journey. The set consists of following 5 states:

- **Offline**: The state corresponds to the time where the vehicle is yet to receive the update either from RSU or one of the other vehicles. The transition to other state occurs only when it has received the complete update
- **Wait**: The state corresponds to the time where it has received the update but does not start acting as source of update. This state is used to avoid contention for channel when one already knows an existence of another vehicle in proximity which can update vehicles. The transition to other states depends on the event occurring due to timeout of different timers.

- **Broadcast**: In this state the vehicle broadcasts the update in its vicinity and transitions occur on events of timeouts
- **Inactive**: A vehicle performs updates with a duty cycle where this state represents the off period while the state of Broadcast represents the on period for broadcast.
- **End**: This state represents the end of trip after which it is unable to participate in communication.

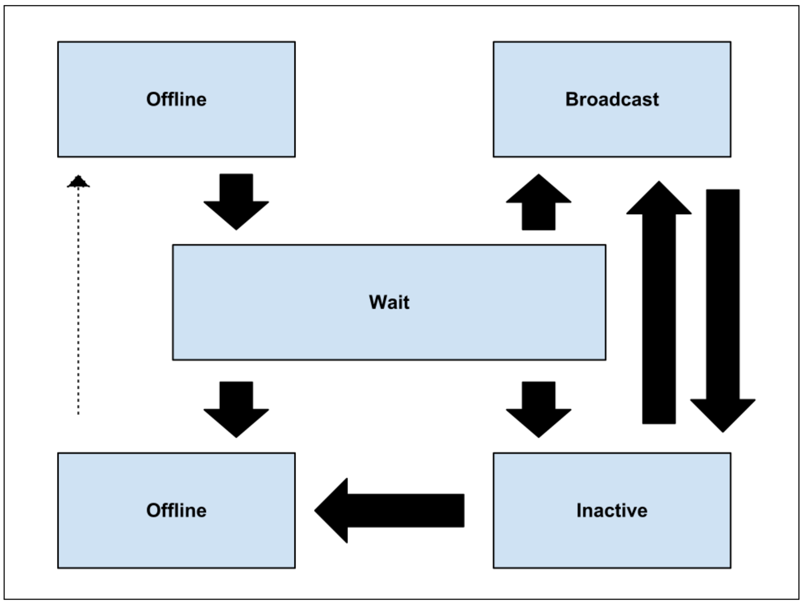


Figure 7-6: Communication life cycle of vehicles for application of Map updates

The flow diagram between the states is shown if Fig 7-4-3. The diagram shows possible transitions between states. Based on these states, a timing diagram is shown in Fig 7-4-3. It shows the various parameters associated with the algorithm. They are: T_{update} represents the time at which vehicle gets updated, $T_d(t)$ represents the time interval between the vehicle broadcasting its 1^{st} update from the time it received the new map data; $N_b(t)$ represents the number of broadcasts, a vehicle will perform in bursts with T_b as the time required to broadcast one update; $N_c(t)$ represents the number of cycles of bursts of broadcasts a vehicle will perform over its entire duration; $T_o(t)$ represents the off period between each of the broadcast burst cycles and finally $T_p(t)$ represents period of broadcast cycle.

7-4-4 Kernel Function

$$K(t) = \begin{cases} 0 & \text{if } t \leq T_{update} \\ 0 & \text{if } t > T_{update} \wedge t \leq T_s \\ m_b(t - T_s + x(T_p) + T_b) & \text{if } t > T_s \wedge t \leq (T_s + x(T_p) + T_b) \\ -m_i(T_s + (x+1)(T_p) - t) & \text{if } t > (T_s + x(T_p) + T_b) \wedge t \leq (T_s + (x+1)(T_p)) \end{cases} \tag{7-1}$$

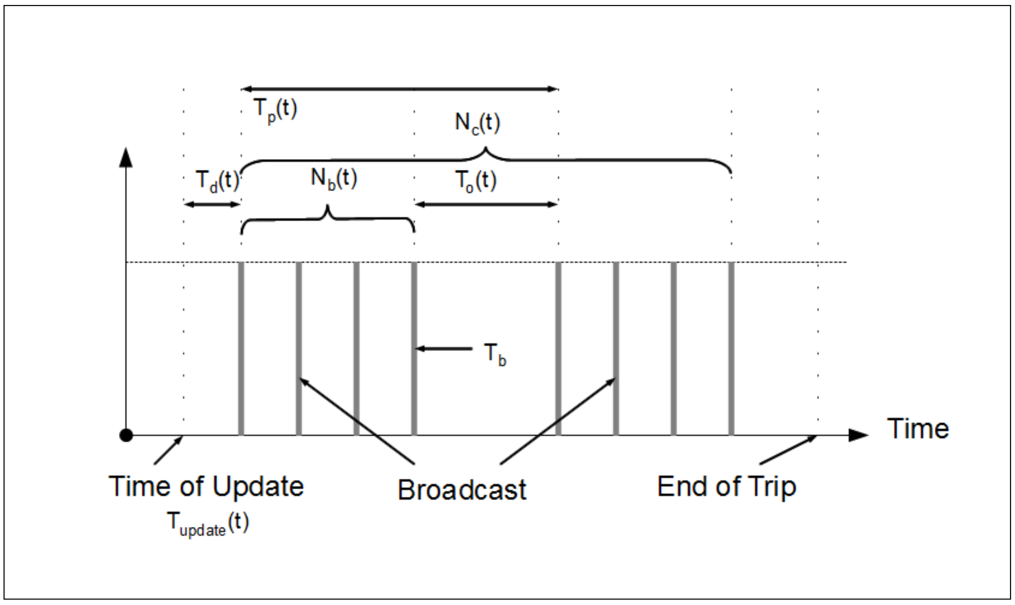


Figure 7-7: Parameters of broadcast for each vehicle

The kernel function characterizes the pattern of updates issued by an updated vehicle. It is a triangular wave whose duty cycle is adaptive to the events of redundant messages or inactivity timeouts. It is represented in equation Eq. 7-1 where $m_b = \frac{1}{T_b}$ and $m_i = \frac{1}{T_i}$ and $T_s = T_{update} + T_d$. The duration for which, the kernel function transitions from $0 \rightarrow 1$ is considered the time in which a vehicle broadcasts. The graphical representation of the kernel function is shown in Fig. 7-4-4 with its algorithm specified in Alg. 7-8

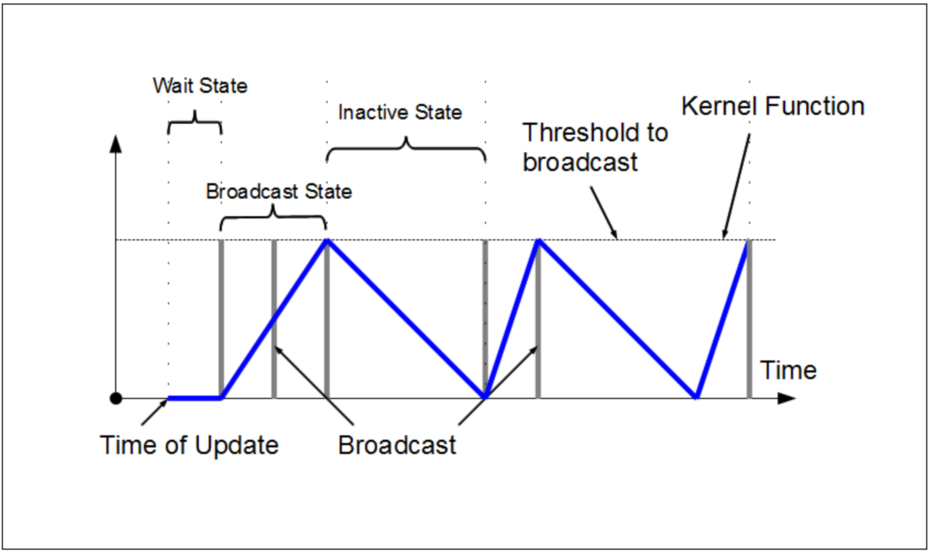


Figure 7-8: Kernel Function and communication states of a vehicle

On an event such as redundant message and timeout of inactivity, the kernel function is modified by changing duty cycle by value of $offset$. This functionality is explained in algorithms specified in Alg.7-8 and 7-8. Further, for too many redundant messages in frequent duration indicates too many vehicles with broadcasting capabilities in proximity. Hence, to avoid unnecessary broadcast during that duration, duty cycle is changed to minimum. This functionality is specified in Alg. 7-8. On the other hand, cases where no activity is heard from nearby vehicles, duty cycle is increased so as to maximize spread of updates. This functionality is explained in Alg. 7-8. Fig 7-4-4 is an example showcasing the adaptiveness of a kernel function for a vehicle when another vehicle with the same broadcast comes in proximity.

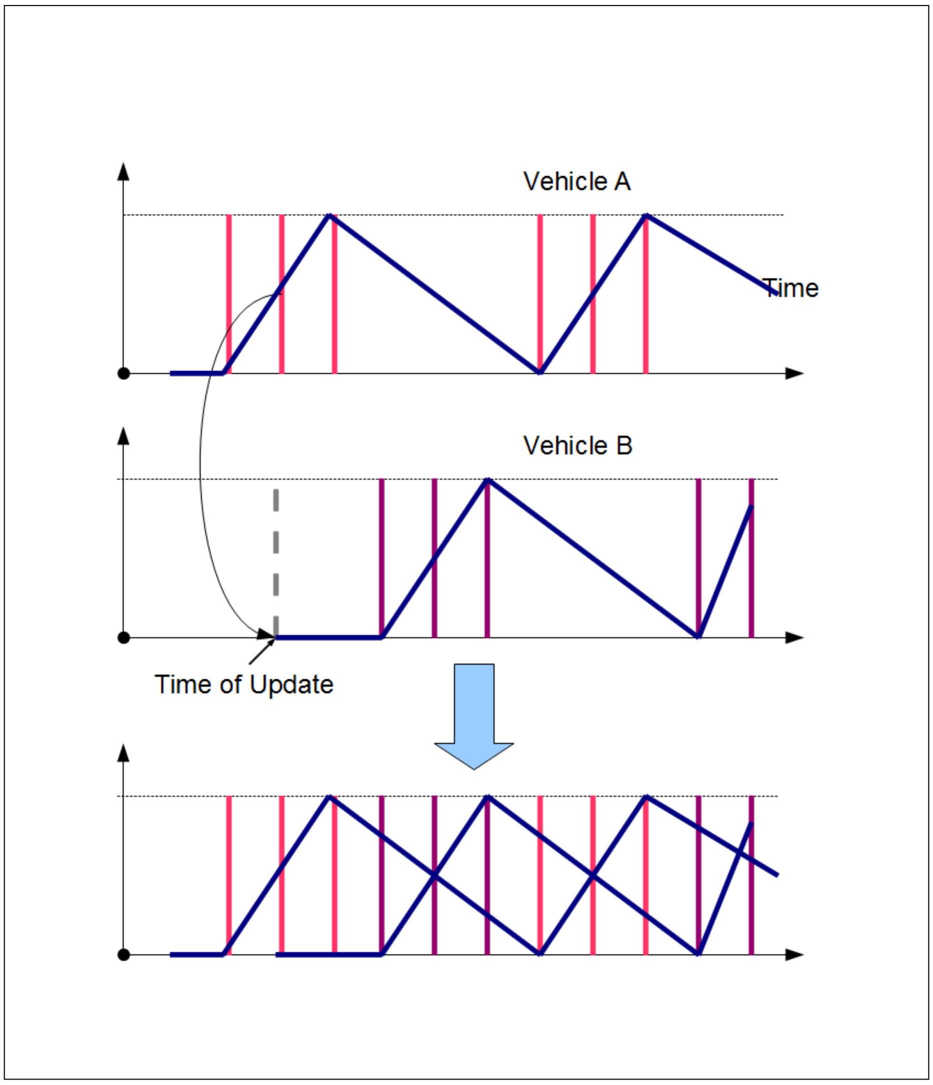


Figure 7-9: Operation of Kernel Function in two vehicle setup

7-5 Simulation

To test the performance Adaptive and Distributive Broadcasting algorithm (ADB) we use our simulator with the configurations as mentioned in the Table 7-5. We test it for 3 different configurations of $offset$ to understand the influence of the parameter on

Communication Range	350 meters
Data rate	24 Mbits/sec
Map Change	120 Mbits
Cars	94000
Simulation Time	24Hrs
Number of Road Side Units	1
Time of Broadcast	2AM
Duration of Broadcast	24 hours
Offset	10, 20 and 30

Table 7-6: Simulation Setup

Offset Value	Spread of Update	Redundant updates	Collisions
10	0.789	4840450	508717
20	0.791	4464561	4933990
30	0.793	3541281	5018593

Table 7-7: Perfromance of ADB algorithm for different $offset$ values

the performance of the system. We perform comparison on three metrics: fraction of traffic updated, number of redundant messages and number of collisions.

7-6 Results

Results for Adaptive and Distributive Broadcasting algorithm (ADB) algorithm are shown in Table 7-6. It shows that the influence of the $offset$ value influences spread of updates in a very marginal with only 0.5% increase seen with increase in $offset$ value. Similarly offset values have no impact on number of collision. Compared to collisions and spread of updates, the influence of offset is high for number of redundant messages. The results show that as values of $offset$ increase so reduces the number of redundant message.

Further, the figures 7-6, 7-6 and 7-6 representing rate of spread of updates, rate of redundant messages and rate of collisions show the same behavior where the offset has no impact on either the spread of updates nor collisions but has significant impact on rate of redundant messages.

To compare the performance of our algorithm ADB we plot redundant messages v/s spread of updates and collision v/s spread of updates. The results are compared with various algorithms based on system parameters and also standard mobility based algorithm of fixed duration of broadcast. The results for redundant messages and collisions are shown in Fig. 7-6 and 7-6 respectively. The results show the superiority of Adaptive and Distributive Broadcasting algorithm (ADB) with being able to reducing redundant messages by 92% as that seen in flooding algorithm with only degradation of 3%. The

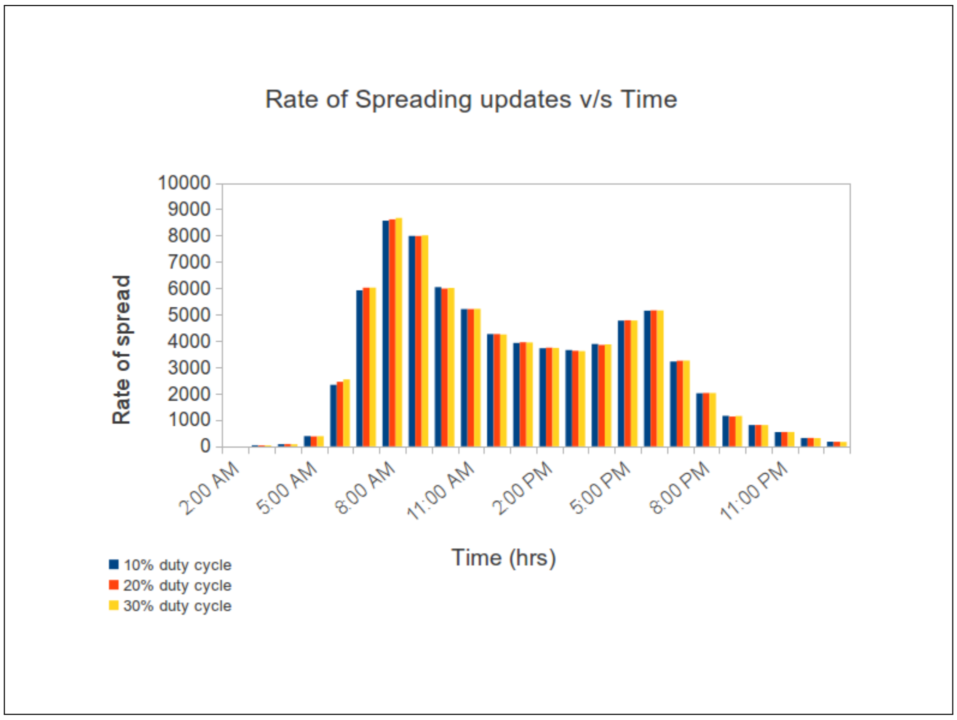


Figure 7-10: Rate of Spread with ADB algorithm

improvements in collisions are observed m, but are not to the extent of that seen with redundant messages. We believe this happens due to the excessive reduction of redundant messages; very less redundant information is shared across vehicles which in turn reduces the ability of vehicles to predict broadcasting cycles of vehicles in its vicinity.

7-7 Conclusion

In this chapter we showcased our distributed Algorithm Adaptive and Distributive Broadcasting algorithm (ADB). It works well to provide spread similar to that of flooding algorithm without the explosion in number of redundant messages and collisions. This is achieved due to the fact that our algorithm being able to locally adapt to its surroundings without affecting other parts of the network. To conclude, our Algorithm ADB performs extremely well by achieving nearly 90% reduction of redundant messages and about 29% of collisions at the mere cost of degradation of 3% in spread of updates as compared to that of flooding algorithm.

7-8 Algorithms

Algorithm 8 Handling of redundant data (redundantReceived(int time))

```
Require: $time \neq 0, state \neq OFFLINE, offset \neq 0$
  $time \leftarrow lastReceivedTime$
  if $state = WAIT$ then
    $T_d \leftarrow T_d + offset$
  else if $state = BROADCAST$ then
    $T_b \leftarrow T_b - offset$
  else if $state = INACTIVE$ then
    $T_o \leftarrow T_o + offset$
  end if
```

Algorithm 9 Handling of inactivity (inactivityReceived(int time))

```
Require: $time \neq 0, state \neq OFFLINE, offset \neq 0,$
  if $(time - lastReceivedTime) \geq (T_b * T_p)$ then
    $T_b \leftarrow T_b + offset$
    $T_o \leftarrow T_o - offset$
    $time \leftarrow lastReceivedTime$
  end if
```

Algorithm 10 Handling of overactivity (overactivity(int time))

```
Require: $time \neq 0, state \neq OFFLINE, offset \neq 0,$
  if $(time - lastReceivedTime) \leq (T_b * T_p)$ then
    $T_b \leftarrow T_b - offset$
    $T_o \leftarrow T_o + offset$
    $time \leftarrow lastReceivedTime$
  end if
```

Algorithm 11 Kernel Function (stateMachine(int time))

```
Require: $time \neq 0, offset \neq 0,$
  if $state = OFFLINE$ then
    if $update = true$ then
      $state \leftarrow WAIT$
    end if
  else if $state = WAIT$ then
    $T_d \leftarrow T_d - 1$
    if $T_d \leq 0$ then
      $state \leftarrow BROADCAST$
    end if
  else if $state = BROADCAST$ then
    $T_b \leftarrow T_b - 1$
    if $T_b \leq 0$ then
      $state \leftarrow INACTIVE$
    end if
  else if $state = INACTIVE$ then
    $T_o \leftarrow T_o - 1$
    if $T_o \leq 0$ then
      $state \leftarrow BROADCAST$
    end if
  else if $state = END$ then
    $exit$
  end if
```

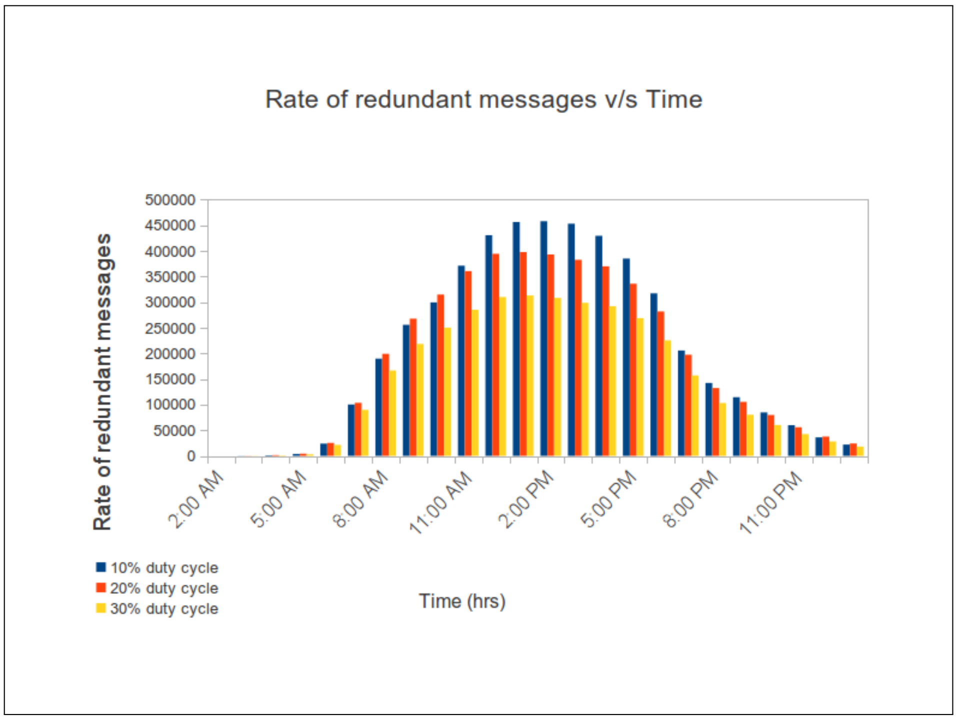


Figure 7-11: Rate of Redundant Messages with ADB algorithm

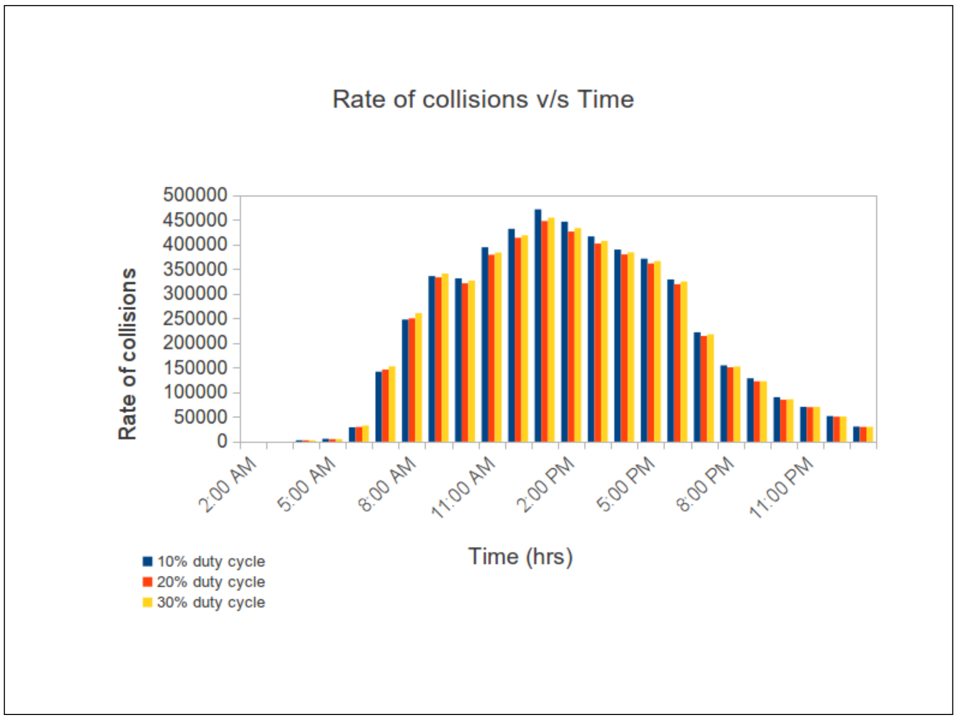


Figure 7-12: Rate of Collisions with ADB algorithm

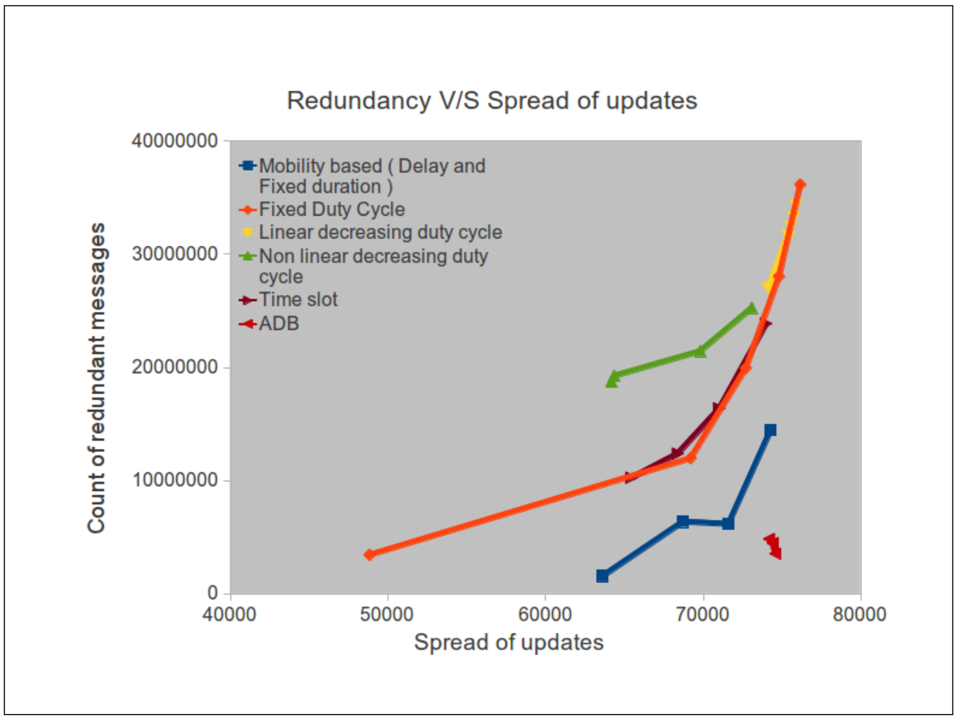


Figure 7-13: Comparision of redundant messages v/s spread of updates for different algorithms

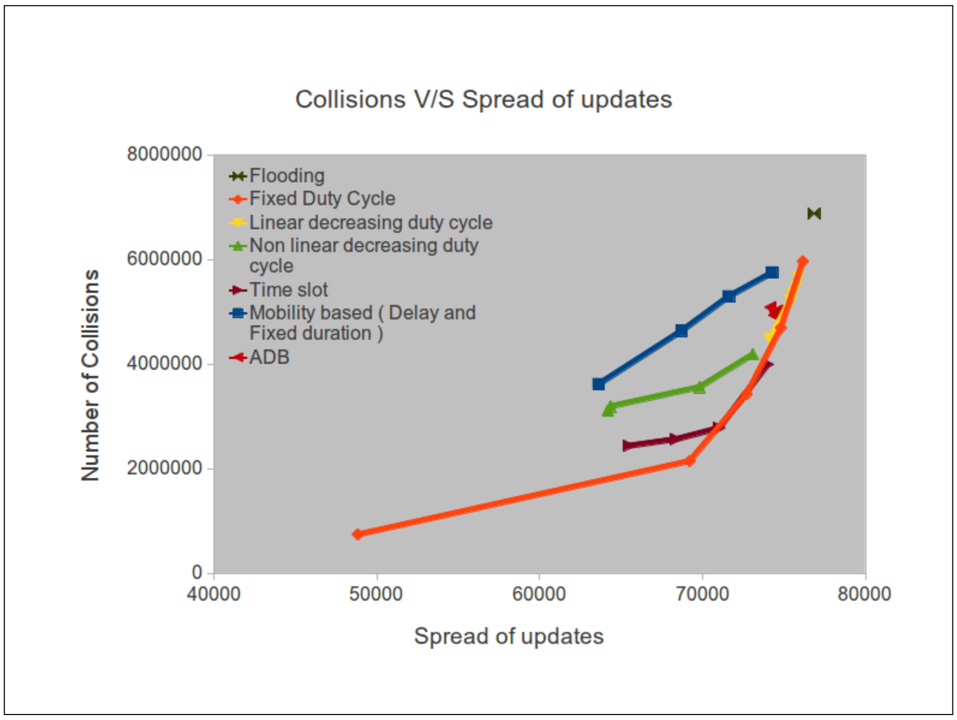


Figure 7-14: Comparision of number of collisions v/s spread of updates for different algorithms

Chapter 8

Conclusion

In this study we investigated the application of map updates in VANET involving various aspects associated with implementation of the application. Requirements of data sizes, type of updates involved in the application, different type of sources, investigation of influences of various parameters of Road Side Unit (RSU) on spread of updates, dissemination algorithm are analyzed in detail. Based on our investigation, assisted with the help of a new, unique simulator developed, we have shown that the map update application is very much feasible to be deployed in VANET. The new simulator, 'DC-PIC' is based on concepts of Divide and Conquer and Particle in Cell. Using this simulator, we have sucessfully simulated extremely large number of mobility traces on standard desktop machines eliminating need of advanced high level computation systems. The simulation results show that wireless map updating of greater than 75% of entire set of vehicles which travel in and around the country of Netherlands over 24 hours is possible. The spread of updates is achieved with a minimal support of RSU demostrating the cost effectiveness of application deployment as weall the ability to deploy it during the early stages of RSU development.

As a part of investigation the author concludes that

- Every location provides different capabilities for updating vehicles.
- Based on the criteria of selecting locations which provide opportunity to update maximum set of vehicles, the highway junctions are the most favorable among all possible locations
- The number of units need not be excessive as even with finite number of units, one can achieve maximum spread.
- Increasing the number of units does not provide any substantial improvement in the increase of number of updates. The increase in the number provides improvement in rate of updates only for initial few hours of the day and after which the

system responds in a similar fashion irrespective of number of units used. The only advantage of increasing the number is seen in improved average time a vehicle requires to get updated.

- In order to achieve maximum spread it is crucial to have time of update by an RSU be early hours of the day. This is due to the fact that vehicles themselves are responsible for spreading updates and so as to be capable to update maximum set of vehicles it is crucial to achieve high count of updated vehicles during the morning rush hour traffic.
- The parameter, duration of update for RSU reveals that it has no significance on spread of updates with a 100% penetration of the technology among vehicles. But its significance grows with lower penetration levels, implying that during early stages of deployment of technology, it would be necessary to provide updates for longer duration and which can be later decreased when the penetration reaches 100%.

Furthermore, we proposed a simple, effective, distributive algorithm Adaptive and Distributive Broadcasting algorithm (ADB) which is based on variable duty cycle of broadcasts with variations occuring with the help of our pseudo-feedback mechanisms. While more refinements in the algorithm possible, the current system is capable to provide about the near same spread as that of flooding algorithm but with extremely low count of redundant messages and collisions.

Chapter 9

Future Work

In this chapter we would like to propose some future research directions for making the the application of Map Updates a reality. We also propose some suggestions with regards to the future support of the tool and the use of mobility traces.

Having shown by simulation that the application of Map updates is possible to provide updates to considerable fraction of traffic, the next logical step would be to test in real world. Simulation has considered to have the communication range at fixed value , which in real world is not at all possible due to the effects of natural or man made structures. This study would provide an practical feasibility of the application. Further, to realize this application one needs to investigate the security breaches that can hamper valid map updates and research on the techniques to overcome these problems.

One of the major finding of the thesis is that, it requires a very small set of vehicles as primary source of update, to obtain a high degree of spread of updates as that one could achieve with the help of RSU. We propose to gather and analyze more information about these set of vehicles which includes number of vehicles, start time of travel, distance of travel etc. This information can be then used to design a model which can in approximation predict the behavior of spreading update. This can then be used to realize an application based on infrastructure less system.

Our feasibility study of map updates relies on using simple 'Tile' based updates and not 'Incremental' updates because of associated complexities. But as incremental updates have an advantage of requiring lesser amount of data as compared to that of Tile based updates, we propose to investigate the changes required to the system and to the dissemination algorithm to implement the application with incremental updates . The purpose of doing so would provide us an comparative study between both modes of updates. Further, such a system would help us generate metrics with which we can find which system performs better under which metrics which can be driven to develop a system and an algorithm which adapt at run time so as to obtain advantages of both the modes.

Map changes occur due to different reasons which include major ones such as blockades on highways while minor ones such as name changes to POIs. All these changes are not of the same type and hence do not require the same spread of updates. The changes to POIs need not be broadcasted to entire country whereas changes to highways do. Similarly changes to highways must be issued at a faster rate than changes to POI. Hence we propose to investigate the changes required in dissemination algorithm VANETs to meet these different requirements.

Map changes consist of different types of changes with major ones such as blockades on highways while minor ones such as name changes to POIs. Changes to POIs do not need to be broadcasted to all part of the country at the earliest whereas the changes to highways would enforce they be issued all across the county and therefore we propose to have map changes be assigned priories and have different characteristics of dissemination of map updates based on them so as to suit the type of changes.

Traffic updates are provided by TomTom as a add-on service with the help of 3G network where every update has a cost involved with the data exchange.We propose to investigate a system consisting of both 3G and IEEE 802.11p based network where the vehicle can verify the presence of update with its neighboring vehicles using IEEE 802.11p and only when it does not request it over 3G. Such a system would help not only to provide updates quickly but also reduce the operating cost of issuing the updates.

The traces and the simulator can be further be used to work on a larger scale such as country as big as Germany or even the entire continent of Europe. Simulations runs with these setups should be able to provide information about similarities and differences in performance for spread of updates in different regions.

Appendix - A

This section contains the source code and scrip files used for simulation.

A-1 Mobility Analyzer

A-1-1 mobilityAnalysis.sh

```
#!/bin/bash

START=$(date +%s)
time=$(date)
clear
echo "###############################################"
echo "Mobility Analyser for DC-PIC Simulator"
echo "Version 1.0"
echo "Date $time"
echo "###############################################"

# validates existing of File simulation.parameters
if [ ! -f simulation.parameters ];
then
  echo ""
  echo "File 'simulation.parameters' does not exist !!"
  echo "Exiting......"
  echo ""
  exit

fi

# Read the parameters from simulation.parameters
toFetchPath=$( gawk '{ if(NR == 1) { print $1 } }' simulation.parameters
   )
```

```
toDestPath=$( gawk  '{ if(NR == 2) { print $1 } }' simulation.parameters
    )
gridSize=$( gawk    '{ if(NR == 3) { print $1 } }' simulation.parameters
    )
timeSlot=$( gawk    '{ if(NR == 4) { print $1 } }' simulation.parameters
    )


# Validate the parameters
if [ -n "$toFetchPath" ] && [ -n "$toDestPath" ] && [ -d $toFetchPath ]
   && [ -d $toDestPath  ]
then
  echo  "Paramters are "
  echo  " Path to read  :$toFetchPath"
  echo  " Path to store :$toDestPath"
  echo "################################################"
else
  echo ""
  echo "Paramters in the file simulation.parameters are empty or
      directories don't exist !! "
  echo "Exiting......"
  echo ""
  exit

fi


# Validate the required Shell Scripts exists exist in the right path
if [ -f generate1MinFiles.sh ] && [ -f sortAndMerge1MinFiles.sh ] && [ -f
     mergedEntries.sh ] && [ -f findLink.sh ]
then
  echo ""
  echo  " Verified existance of required Shell scripts .... "
  echo ""
else
  echo ""
  echo "One of SHELL scrits (generate1MinFiles.sh/sortAndMerge1MinFiles.
      sh/mergedEntries.sh/findLinks.sh ) does NOT EXIST !!!"
  echo "Exiting......"
  echo ""
  exit

fi

# Validate the required AWK scripts exists in the right path
if [ -f mergedEntries_filterData.awk ] && [ -f findLink_merge1SecLinks.
    awk ] && [ -f findLink_splitLinks.awk ]
then
  echo ""
  echo  " Verified existance of required AWK scripts .... "
  echo ""
else
  echo ""
  echo "One of AWK scrits (mergedEntries_filterData.awk/
     findLink_merge1SecLinks.awk/findLink_splitLinks.awk ) does NOT EXIST
      !!!"
  echo "Exiting......"
  echo ""
  exit

fi


# Validate the required jar file exists in the right path
if [ -f FastAlgo.jar ]
then
  echo ""
  echo  " Verified existance of required jar files .... "
  echo ""
else
  echo ""
  echo "One of jar files (FastAlgo.jar  ) does NOT EXIST !!!"
  echo "Exiting......"
  echo ""
  exit

fi



rm -f command.cmd




# Validates whether the required step has already been previously
    executed if not then
# generates files sorted into 1440 files representing for every minute
    for 24 hours for
# cars with different starting number;

# The purpose of doing so was was to split the work and perfom the netire
     process in
# small steps.
if [ -d $toDestPath/fast1/ ];
then
  echo ""
  echo " Step 1: Already generated 1 Minute Files ............., so
     moving on"
  echo ""
else
  echo ""
  echo " Step 1: Generating 1 Minute Files"
  echo ""

    for i in {0..2}
    do
      echo starting with $i

      val1=$((($i*3)+1))
      val2=$((($i*3)+2))
      val3=$((($i*3)+3))

      echo ./generate1MinFiles.sh $val1 $toFetchPath $toDestPath >>
          command.cmd
      echo ./generate1MinFiles.sh $val2 $toFetchPath $toDestPath >>
          command.cmd
      echo ./generate1MinFiles.sh $val3 $toFetchPath $toDestPath >>
          command.cmd
    done

    ppss -f command.cmd -c 'bash $ITEM' -p 3

  echo ""
  echo " Completed 1 Minute Files"
  echo ""
fi



rm -f command.cmd

# Validates whether the required step has already been previously
    executed if not then
# merges the files from the 9 groups one single group containing 1440
    files sorted by time
if [ -d $toDestPath/merge ];
then
  echo ""
  echo " Step 2: Already Merged folder exists ............., so moving on
      "
  echo ""
else
  echo ""
  echo " Step 2: Generating Merged Files"
  echo ""
    rm -rf $toDestPath/merge

    mkdir $toDestPath/merge

    for i in {0..479}
    do

      val1=$((($i*3)+0))
      val2=$((($i*3)+1))
      val3=$((($i*3)+2))

      echo ./sortAndMerge1MinFiles.sh $val1 $toDestPath >> command.cmd
      echo ./sortAndMerge1MinFiles.sh $val2 $toDestPath >> command.cmd
      echo ./sortAndMerge1MinFiles.sh $val3 $toDestPath >> command.cmd
    done

    ppss -f command.cmd -c 'bash $ITEM' -p 5

  echo ""
  echo " Completed Generating Merged Files"
  echo ""
fi




# Validates whether the required step has already been previously
    executed if not then
# executing through all 1440 files assign them into grids based on size
    mentioned in
# 'simulation.parameters' as the 3rd parmaters

j=$gridSize

actualSize=$(( ( $gridSize + 1) *10 ))

rm -f command.cmd

if [ -d $toDestPath/match_$j ];
then
  echo ""
  echo " Step 3: Already matched entries exists for $actualSize meters
      ............., so moving on"
  echo ""
else
  echo ""
  echo " Step 3: Generating matched entry Files for $actualSize meters"
  echo ""

  rm -rf $toDestPath/match_$j


  rm -f command.cmd
  mkdir $toDestPath/match_$j

  for i in {0..479}
  do
    val1=$((($i*3)+0))
    val2=$((($i*3)+1))
    val3=$((($i*3)+2))

    echo ./mergedEntries.sh $val1 $toDestPath $j >> command.cmd
    echo ./mergedEntries.sh $val2 $toDestPath $j >> command.cmd
    echo ./mergedEntries.sh $val3 $toDestPath $j >> command.cmd
  done

  ppss -f command.cmd -c 'bash $ITEM' -p 5

  echo ""
  echo " Completed matched entry Files"
  echo ""
fi


# Validates whether the required step has already been previously
    executed if not then
# executing through all 1440 files based on grids finds the potential
    links between cars.

j=$gridSize

if [ -f $toDestPath/match_$j/test.test ];
then
  echo ""
  echo " Step 4: Already links calculated for $actualSize meters
      ............., so moving on"
  echo ""
else
  echo ""
  echo " Step 4: Generating links entry Files for $actualSize meters"
  echo ""

  ./findLink.sh $toDestPath $j $timeSlot

  echo ""
  echo " Completed created list of potential Links"
  echo ""
fi



END=$(date +%s)

DIFF=$(( $END - $START ))

size=$( stat -c %s $toDestPath/match_$j/$j.$timeSlot.link )

if [ $size -ne 0 ];
then
  echo "
      ***********************************************************************
      "
  echo " File of Potential Links for Grid $actualSize X $actualSize m2
      generated in $DIFF seconds!!!!!"
  echo " File Name '$j.$timeSlot.link' "
  echo " File Path '$toDestPath/match_$j/' "
  echo "
      ***********************************************************************
      "
  echo ""
  echo ""
  echo ""
else
  echo "File generated is empty !!!!!!"
fi
```

A-1-2 generate1MinFiles.sh

```
#!/bin/bash

cd $2

ls t_trip$1* > ../list.txt_$1

num=$(ls t_trip$1* | wc -l)

rm -rf fast_$1

mkdir fast_$1

java -jar "/home/pavan/NetBeansProjects/FastAlgo/dist/FastAlgo.jar" 0 ../
    list.txt_$1 $1 > $1.tmp

cd fast_$1/

rm -rf $3/fast$1/
mkdir $3/fast$1/

for file in `dir -d *.1minFile` ; do

  sort -k1n -k5n,6n $file > sort_$file

  mv sort_$file $3/fast$1/

done
```

A-1-3 sortAndMerge1MinFiles.sh

```
#!/bin/bash

cd $2

for i in {1..9}
do

  cat fast$i/sort_$1.* >> merged_$1.dat

done

  sort -k1n -k5n,6n merged_$1.dat | uniq > sort_merged_$1.dat

  rm merged_$1.dat

  mv sort_merged_$1.dat merge/
```

A-1-4 findLink.sh

```
#!/bin/bash

#cd $1

#cd match_$2

rm -f $1/match_$2/test.test

for i in {0..1439}
do
  echo $i
  cat $1/match_$2/pairDatatemp_$i.dat_$2 >> $1/match_$2/test.test

done

echo "A) Finished merging all the files"
cat $1/match_$2/test.test | sort -k4n -k5n -k1n -k2n -k3n > $1/match_$2/
    tmp.tmp

echo "B) Finished sorting the merged file"
gawk -f findLink_merge1SecLinks.awk $1/match_$2/tmp.tmp > $1/match_$2/tmp
    .tmp2

  i=$3
  echo "step C"
  gawk -v time=$i '{ if ( $6 >= time ) { if ( ( time * ( int($1/time) ) )
      == $1 ) { print $0, time * ( int($1/time) ) } else { print $0, time
      * ( ( int($1/time) ) + 1 ) } } }'  $1/match_$2/tmp.tmp2 > $1/
     match_$2/tmp.tmp3

  echo "step D"
  gawk -v time=$i '{ if ( ( ( $1 + $6 ) - $7 ) >= time ) { print $7, $2,
     $3, $4, $5, ( ( $1 + $6 ) - $7 ) } }' $1/match_$2/tmp.tmp3 > $1/
     match_$2/tmp.tmp4

  echo "step E"
  gawk -v value=$i -f findLink_splitLinks.awk $1/match_$2/tmp.tmp4 > $1/
     match_$2/tmp.tmp5

  echo "step F"
  cat $1/match_$2/tmp.tmp5 | sort -k1n -k4n -k5n -k2n -k3n > $1/match_$2/
     $2.$i.link
```

A-1-5 mergedEntries.sh

```
#!/bin/bash
```

```

cd $2/merge

value=$(($3+1))

gawk -v val=$value -f mergedEntries_filterData.awk sort_merged_$1.dat >
    temp_$1.dat


java -jar "/home/pavan/NetBeansProjects/FastAlgo/dist/FastAlgo.jar" 2
    temp_$1.dat $3

rm temp_$1.dat
```

A-1-6 findLink-merge1SecLinks.awk

```
BEGIN {

time = 0
lat  = 0
lng  = 0
car1 = 0
car2 = 0

currTime = 0

}
{

  if ( NR == 1 ) {

time = $1
lat  = $2
lng  = $3
car1 = $4
car2 = $5
currTime = time + 1
  } else {

    if ( car1 == $4 && car2 == $5 && lat == $2 && lng == $3 ) {

      currTime = $1 + 1

    } else {

print time, lat, lng, car1, car2, ( currTime - time )

time = $1
lat  = $2
lng  = $3
car1 = $4
car2 = $5
currTime = time + 1
```

```
      }

    }



}
END {

print time , lat , lng , car1 , car2 , ( currTime - time )

}
```

A-1-7 otherCarCount-merge.awk

```
BEGIN {

time = 0
interTime = 0

car1 = 0
car2 = 0



}
{

    if ( NR == 1 ) {

time = $1
interTime = $1

car1 = $4
car2 = $5


    } else {

      if ( car1 == $4 && car2 == $5 ) {

        if ( ( $1 - interTime ) == 1 ) {

interTime = $1

        } else {

          diff = interTime - time + 1

          print time , interTime , diff , car1 , car2

time = $1
interTime = $1
```

```

      }

    } else {

        diff = interTime - time + 1

        print time, interTime, diff, car1, car2



time = $1
interTime = $1

car1 = $4
car2 = $5

    }

  }

}
END {

  diff = interTime - time + 1

  print time, interTime, diff, car1, car2

}
```

A-1-8 findLink-splitLinks.awk

```
BEGIN {

time = 0
lat  = 0
lng  = 0
car1 = 0
car2 = 0

currTime = 0

}
{

time     = $1
lat      = $2
lng      = $3
car1     = $4
car2     = $5
period   = $6


if ( period >= value )
{

  for ( ind = time ; ind < ( ( time + period ) + 1 ) ; ind = ind + value
     )
  {

    print ind, lat, lng, car1, car2

  }

}



}
END {


}
```

A-1-9 mergedEntries-filterData.awk

```
BEGIN {

        oldTime =  0
        oldLatId=  0
        oldLngId=  0
        carId =  0

        count = 0

}
{

  if ( NR == 1 ) {

        oldTime =  $1
        oldLatId=  int( $5 / val )
        oldLngId=  int( $6 / val )
        carId =  $2
        count = 0



  } else {


    if ( oldTime == $1 && oldLatId == int( $5 / val ) && oldLngId == int(
        $6 / val ) ) {

        #if ( NR == 2 ) {

```

```
        # print oldTime, oldLatId, oldLngId, carId

        #}

        print oldTime, carId, oldLatId, oldLngId

        #print $1, $2, $5, $6

        count = 1

    } else {

      if ( count == 1 ) {

        print oldTime, carId, oldLatId, oldLngId

      }

        count = 0
    }


        oldTime =  $1
        oldLatId=  int( $5 / val )
        oldLngId=  int( $6 / val )
        carId =  $2


  }

  #print NR

}
END {

      if ( count == 1 ) {

        print oldTime, carId, oldLatId, oldLngId

      }

}
```

A-1-10 simulation.parameters

```
/home/pavan/Thesis/traces/0108/firstAug     - File Path to read Mobility
    traces
/media/DATAPART1/TraceData/fastDataNew      - File Path to to store
    traces
49                 - Size of Grid ( ( value + 1 ) * 10 )
1                - Link existance for number of seconds
50.744695198751           - Lower Latitude of Area under Consideration
54.626505724435           - Upper Latitude of Area under Consideration
```

```
3.3099518114947          — Left Longitude of Area under Consideration
7.2375641167286          — Right Longitude of Area under Consideration
1312156800           — Start Time of Traces
86400              — Duration of Traces
```

A-2 Road Side Unit - Generation

A-2-1 rsuCreator.sh

```
#!/bin/bash

###########################################################
#
# $1 = /media/DATAPART1/TraceData/fastDataNew/merge
# $2 = /media/DATAPART1/TraceData/fastDataNew/Infrastructure/Time_
# $3 = 50 ( for 500m as communication range )
# $4 = 5 ( time duration )
# $5 = 100 Max Entries
#
###########################################################

START=$(date +%s)
time=$(date)
clear
echo "#############################################"
echo "RSU Generator DC-PIC Simulator"
echo "Version 1.0"
echo "Date $time"
echo "#############################################"


if [ ! -f simulation.parameters ];
then
  echo ""
  echo "File 'simulation.parameters' does not exist !!"
  echo "Exiting......"
  echo ""
  exit

fi

# Read the parameters from simulation.parameters
toFetchPath=$( gawk '{ if(NR == 1) { print $1 } }' simulation.parameters
    )
toTempPath=$( gawk  '{ if(NR == 2) { print $1 } }' simulation.parameters
    )
gridSize=$( gawk    '{ if(NR == 3) { print $1 } }' simulation.parameters
    )
timeSlot=$( gawk    '{ if(NR == 4) { print $1 } }' simulation.parameters
    )
numOfEntries=$( gawk    '{ if(NR == 5) { print $1 } }' simulation.
    parameters )
```

```
actualSize=$(( ( $gridSize ) *10 ))
toDestPath=$toTempPath$timeSlot

# Validate the parameters
if [ -n "$toFetchPath" ] && [ -n "$toDestPath" ] && [ -d $toFetchPath ]
then
  echo  "Paramters are "
  echo  " Path to read  :$toFetchPath"
  echo  " Path to store :$toDestPath"
  echo "################################################"
else
  echo ""
  echo "Paramters in the file simulation.parameters are empty or
     directories don't exist !! "
  echo "Exiting......"
  echo ""
  exit

fi

# Validate the required Shell Scripts exists exist in the right path
if [ -f findLatLngHistStep1.sh ] && [ -f findLatLngHistStep2.sh ] && [ -f
    findLatLngHistStep3.sh ] && [ -f findLatLngHistStep1_findCount.sh ]
then
  echo ""
  echo  " Verified existance of required Shell scripts .... "
  echo ""
else
  echo ""
  echo "One of SHELL scrits (findLatLngHistStep1.sh/findLatLngHistStep2.
     sh/findLatLngHistStep3.sh ) does NOT EXIST !!!"
  echo "Exiting......"
  echo ""
  exit

fi

# Validate the required AWK Scripts exists exist in the right path
if [ -f findLatLngHistStep1_findCount_merge.awk ] && [ -f
    findLatLngHistStep3_merge.awk ]
then
  echo ""
  echo  " Verified existance of required AWK scripts .... "
  echo ""
else
  echo ""
  echo "One of AWK scrits (findLatLngHistStep1_findCount_merge.awk/
     findLatLngHistStep3_merge.awk/findLatLngHistStep3.sh ) does NOT
     EXIST !!!"
  echo "Exiting......"
  echo ""
  exit

```

```
fi

if [ -d $toDestPath ]
then
  echo ""
  echo " Step 1: Already Calculated  Number of Cars associated with Grid
      for every minute"
  echo ""
else
  echo ""
  echo " Step 1: Counting Number of Cars associated with Grid for every
      minute"
  echo ""

  ./findLatLngHistStep1.sh $toFetchPath $gridSize $timeSlot
fi

if [ -f $toDestPath/sortedData.dat.$timeSlot ]
then
  echo ""
  echo " Step 2: Already merged counts over 24 hours and obtained top
      count list"
  echo ""
else
  echo ""
  echo " Step 2: Merging count over 24 hours and obtain top count list"
  echo ""

  ./findLatLngHistStep2.sh $toDestPath $gridSize $timeSlot
fi

if [ -f $toDestPath/timeOfPass_1.dat ]
then
  echo ""
  echo " Step 2: Already generated RSU File"
  echo ""
else
  echo ""
  echo " Step 2: Generating Road Side Unit Files"
  echo ""

  ./findLatLngHistStep3.sh $toDestPath $toFetchPath $numOfEntries
      $gridSize $timeSlot
fi

END=$(date +%s)

DIFF=$(( $END - $START ))

size=$( stat -c %s $toDestPath/timeOfPass_1.dat )

if [ $size -ne 0 ];
then
```

```
  echo "
     **********************************************************************
     "
  echo " File of RSUs with Grid $actualSize X $actualSize m2 generated in
      $DIFF seconds!!!!!"
  echo " File Name 'timeOfPass_X.dat' "
  echo " File Path '$toDestPath' "
  echo "
     **********************************************************************
     "
  echo ""
  echo ""
  echo ""
else
  echo "File generated is empty !!!!!!"
fi
```

A-2-2 findLatLngHistStep1findCount.sh

```
#!/bin/bash

###########################################################
#
# $1 = /media/DATAPART1/TraceData/fastDataNew/merge
# $2 = 50 ( for 500m as communication range )
# $3 = 5 ( time duration )
# $4 = File Id
#
###########################################################

  cd $1

  gawk -v val1=$2 '{  print $1, $2, int($5/val1 ), int($6/val1 ) }' $1/
     sort_merged_$4.dat > $1/temp.latlng_$4.$3

  cat $1/temp.latlng_$4.$3 | sort -k2 -k1n | uniq > $1/temp1.latlng_$4.$3

  gawk -f findLatLngHistStep1_findCount_merge.awk $1/temp1.latlng_$4.$3 >
      $1/temp2.latlng_$4.$3

  gawk -v val2=$3 '{ if ( ( $2 - $1) >= val2 ) { print $3,$4,$5 } }' $1/
     temp2.latlng_$4.$3 > $1/latlng_$2.latlng_data_$4.$3

  mv $1/latlng_$2.latlng_data_$4.$3 $1/../Infrastructure/Time_$3
```

A-2-3 findLatLngHistStep2.sh

```
#!/bin/bash

###########################################################
#
# $1 = /media/DATAPART1/TraceData/fastDataNew/Infrastructure/Time_5
# $2 = 50 ( for 500m as communication range )
# $3 = 5 ( time duration )
#
############################################################



count=0

rm -f $1/mergedLatLng.dat.$3

for j in {0..1439}

do

  cat $1/latlng_$2.latlng_data_$j.$3 >> $1/mergedLatLng.dat.$3

  #echo latlng_$2.latlng_data_$j.$3

done

# This will filter out multiple entries of same cars at the same location
cat $1/mergedLatLng.dat.$3 | sort -k1 -k2n -k3n | uniq > $1/mergedLatLng2
    .dat.$3

# This will give us the count of cars that travelled through locations
gawk '{ print $2, $3 }' $1/mergedLatLng2.dat.$3 > $1/mergedLatLng3.dat.$3
cat $1/mergedLatLng3.dat.$3 | sort -k1n -k2n | uniq -c > $1/mergedLatLng4
    .dat.$3

gawk -v val1=$2 '{ lat= ( ( $2 * val1 * 0.000090 ) + 50.744695198751 );
    lng = ( ( $3 * val1 * 0.000090 ) + 3.3099518114947 ); print $1,lat,lng
    , $2, $3 }' $1/mergedLatLng4.dat.$3 > $1/finalMergedLatLng.dat

#gawk -f /home/pavan/Documents/algorithmn/scripts/latLng/mergeLatLngCount
    .awk mergedLatLng2.dat.$3 > mergedLatLng3.dat.$3

cat $1/finalMergedLatLng.dat | sort -nrk 1 > $1/sortedData.dat.$3
```

A-2-4 findLatLngHistStep3.sh

```
#!/bin/bash

############################################################
#
# $1 = /media/DATAPART1/TraceData/fastDataNew/Infrastructure/Time_5
# $2 = /media/DATAPART1/TraceData/fastDataNew/merge
# $3 = 100 ( Count of entries to fetch )
# $4 = 50  ( for 500m as communication range )
# $5 = 5   ( time duration )
#
############################################################

```

```


gawk line=$3'{ if ( NR <= line ) { print $0 } }' $1/sortedData.dat.$5 > $2/filteredData.$5



count=0

rm -f $2/tmp_*.cars

# Here 20 is to indicate 10Kms block cos the data is already split in 500 m block so only furthe reduction of 200 will fetch us 10Km block
gawk '{ print $1, $2, $3, int($4/20), int ($5/20), $4, $5 }' $2/filteredData.$5 > $2/potensial_$5.points


# Filtering is done for thrice so as to get unique points which are ateast 10Kms apart and the count is near to 5

latValue=$( gawk -v lineNum=1 '{ if ( NR == lineNum ) { print int( $4  )  } }' $2/potensial_$5.points )
lngValue=$( gawk -v lineNum=1 '{ if ( NR == lineNum ) { print int( $5  )  } }' $2/potensial_$5.points )
gawk -v lineNum=1 -v l1=$latValue -v l2=$lngValue '{ if ( NR > lineNum ) {  valx = int( $4  ); valy = int( $5  );  if ( ( ( valx > ( l1 + 1 ) ) || ( valx < ( l1 - 1 ) ) ) && ( ( valy > ( l2 + 1 ) ) || ( valy < ( l2 - 1 ) ) ) ) { print $0 } } else { print $0 } }' $2/potensial_$5.points > $2/potensial_$5.points_1

lineCount=$( gawk 'END{ print NR }' $2/potensial_$5.points_1 )

if [ "$lineCount" -ge 5 ]; then

  latValue=$( gawk -v lineNum=2 '{ if ( NR == lineNum ) { print int( $4  )  } }' $2/potensial_$5.points_1 )
  lngValue=$( gawk -v lineNum=2 '{ if ( NR == lineNum ) { print int( $5  )  } }' $2/potensial_$5.points_1 )
  gawk -v lineNum=2 -v l1=$latValue -v l2=$lngValue '{ if ( NR > lineNum ) {  valx = int( $4  ); valy = int( $5  );  if ( ( ( valx > ( l1 + 1 ) ) || ( valx < ( l1 - 1 ) ) ) && ( ( valy > ( l2 + 1 ) ) || ( valy < ( l2 - 1 ) ) ) ) { print $0 } } else { print $0 } }' $2/potensial_$5.points_1 > $2/potensial_$5.points_2

  lineCount=$( gawk 'END{ print NR }' $2/potensial_$5.points_2 )

  if [ "$lineCount" -ge 5 ]; then

    latValue=$( gawk -v lineNum=3 '{ if ( NR == lineNum ) { print int( $4  )  } }' $2/potensial_$5.points_2 )
    lngValue=$( gawk -v lineNum=3 '{ if ( NR == lineNum ) { print int( $5  )  } }' $2/potensial_$5.points_2 )
    gawk -v lineNum=3 -v l1=$latValue -v l2=$lngValue '{ if ( NR >
        lineNum ) {  valx = int( $4  ); valy = int( $5  );  if ( ( ( valx
        > ( l1 + 1 ) ) || ( valx < ( l1 - 1 ) ) ) && ( ( valy > ( l2 + 1 )
         ) || ( valy < ( l2 - 1 ) ) ) ) { print $0 } } else { print $0 }
        }' $2/potensial_$5.points_2 > $2/potensial_$5.points_3

  fi

fi



lineCount=$( gawk 'END{ print NR }' $2/potensial_$5.points_3 )

if [ "$lineCount" -ge 5 ]; then

lineCount=5

fi


for i in {1..4}
do

  latValue=$( gawk -v lineNum=$i '{ if ( NR == lineNum ) { print $6  } }'
       $2/potensial_$5.points_3 )
  lngValue=$( gawk -v lineNum=$i '{ if ( NR == lineNum ) { print $7  } }'
       $2/potensial_$5.points_3 )

  for j in {0..1439}

  do

    gawk -v val1=$4 -v l1=$latValue -v l2=$lngValue '{ if ( ( int($5/val1
       ) == l1 ) && ( int($6/val1 ) == l2 )  ) { print $1, $2, int($5/
        val1 ), int($6/val1 ) } }' $2/sort_merged_$j.dat >> $2/tmp_$i.cars

    #echo $i $j $latValue $lngValue

  done

  cat $2/tmp_$i.cars | sort -k2 -k1n -k3n -k4n  > $2/tmp2_$i.cars

  gawk -f findLatLngHistStep3_merge.awk $2/tmp2_$i.cars > $2/tmp3_$i.cars

  gawk -v val1=$4 -v val2=$5 '{ if ( ( $2 - $1) >= val2 ) { print $1, $3,
      ( $4 * val1 ), ( $5 * val1), $4, $5 } }' $2/tmp3_$i.cars > $2/
     tmp4_$i.cars

  cat $2/tmp4_$i.cars | sort -k1n -k2 -k3n -k4n > $1/timeOfPass_$i.dat



```

```
done

#echo "The line count is $lineCount"
```

A-2-5 findLatLngHistStep1-findCount-merge.awk

```
BEGIN {

time = 0
endTime = 0
car1 = 0
lat=0
lng=0

}
{

  if ( NR == 1 ) {

time    = $1
endTime = $1
car1    = $2
lat     = $3
lng     = $4


  } else {

    if ( car1 == $2 && lat == $3 && lng == $4 ) {

      if ( ( $1 - endTime ) == 1 ) {

        endTime = $1

      } else {

print time, (endTime + 1 ), car1, lat, lng

        time = $1
        endTime = $1

      }

    } else {

print time, (endTime + 1 ), car1, lat, lng



time    = $1
endTime = $1
car1    = $2
lat     = $3
lng      = $4

    }

  }

}
END {

print time, (endTime + 1 ), car1, lat, lng

}
```

A-2-6 findLatLngHistStep3-merge.awk

```
BEGIN {

time = 0
endTime = 0
car1 = 0
lat=0
lng=0
flat=0
flng=0
}
{

  if ( NR == 1 ) {

time     = $1
endTime = $1
car1     = $2
lat      = $3
lng      = $4


  } else {

    if ( car1 == $2 && lat == $3 && lng == $4 ) {

      if ( ( $1 - endTime ) == 1 ) {

        endTime = $1

      } else {

print time, (endTime + 1 ), car1, lat, lng

        time = $1
        endTime = $1

      }

    } else {

print time, (endTime + 1 ), car1, lat, lng



time    = $1
endTime = $1
car1    = $2
lat     = $3
lng     = $4

    }

  }

}
END {

print time, (endTime + 1 ), car1, lat, lng

}
```

Bibliography

[1] "Car 2 car communication consortium." [Online]. Available: http://www.car-to-car.org/

[2] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "Sumo - simulation of urban mobility: An overview," in *SIMUL 2011, The Third International Conference on Advances in System Simulation*, Barcelona, Spain, October 2011, pp. 63–68.

[3] K. Lee and M. Gerla, "Opportunistic vehicular routing," in *Wireless Conference (EW), 2010 European*, april 2010, pp. 873 –880.

[4] "Google maps : http://maps.google.com/." [Online]. Available: http://maps.google.com/

[5] "Google earth : http://earth.google.com/." [Online]. Available: http://earth.google.com/

[6] "Ieee 802.11p : http://standards.ieee.org/." [Online]. Available: http://standards.ieee.org/findstds/standard/802.11p-2010.html

[7] "Bmw: Digital map changes." [Online]. Available: http://www.bmw.com/com/en/owners/accessories/accessoryfinder/finder.html?proDsId=2369

[8] "Tomtom: Digital map changes." [Online]. Available: http://www.tomtom.com/en_us/maps/map-update-service/

[9] "Bmw: Digital map updates." [Online]. Available: http://www.aqualab.cs.northwestern.edu/projects/STRAW/index.php

[10] "Tomtom:lifetime map updates." [Online]. Available: http://www.tomtom.com/services/service.php?id=51

[11] M. Caliskan, D. Graupner, and M. Mauve, "Decentralized discovery of free parking places," in *Proceedings of the 3rd international workshop on Vehicular ad hoc networks*, ser. VANET '06. New York, NY, USA: ACM, 2006, pp. 30–39. [Online]. Available: http://doi.acm.org/10.1145/1161064.1161070

[12] C. Wewetzer, M. Caliskan, and A. Luebke, "The feasibility of a search engine for metropolitan vehicular ad-hoc networks," in *Globecom Workshops, 2007 IEEE*, nov. 2007, pp. 1 –8.

[13] O. K. Tonguz and M. Boban, "Multiplayer games over vehicular ad hoc networks: A new application," *Ad Hoc Networks*, vol. 8, no. 5, pp. 531 – 543, 2010, <ce:title>Vehicular Networks</ce:title>. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570870509001334

[14] P. Bucciol, F. Ridolfo, and J. De Martin, "Multicast voice transmission over vehicular ad hoc networks: Issues and challenges," in *Networking, 2008. ICN 2008. Seventh International Conference on*, april 2008, pp. 746 –751.

[15] M. Guo, M. Ammar, and E. Zegura, "V3: a vehicle-to-vehicle live video streaming architecture," in *Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on*, march 2005, pp. 171 – 180.

[16] H. Noshadi, E. Giordano, H. Hagopian, G. Pau, M. Gerla, and M. Sarrafzadeh, "Remote medical monitoring through vehicular ad hoc network," in *Vehicular Technology Conference, 2008. VTC 2008-Fall. IEEE 68th*, sept. 2008, pp. 1 –5.

[17] A. Nandan, S. Das, G. Pau, M. Gerla, and M. Sanadidi, "Co-operative downloading in vehicular ad-hoc wireless networks," in *Wireless On-demand Network Systems and Services, 2005. WONS 2005. Second Annual Conference on*, jan. 2005, pp. 32 – 41.

[18] "Ns2 : http://www.isi.edu/nsnam/ns/." [Online]. Available: http://www.isi.edu/nsnam/ns/

[19] "Cmu monarch projects." [Online]. Available: http://www.monarch.cs.rice.edu/cmu-ns.html

[20] "Glomosim : http://pcl.cs.ucla.edu/projects/glomosim/." [Online]. Available: http://pcl.cs.ucla.edu/projects/glomosim/

[21] "Qualnet." [Online]. Available: http://www.qualnet.com/content/

[22] "Jist : http://jist.ece.cornell.edu/." [Online]. Available: http://jist.ece.cornell.edu/

[23] R. Barr, Z. J. Haas, and R. van Renesse, "Jist: an efficient approach to simulation using virtual machines: Research articles," *Softw. Pract. Exper.*, vol. 35, no. 6, pp. 539–576, May 2005. [Online]. Available: http://dx.doi.org/10.1002/spe.v35:6

[24] "Opnet : http://www.opnet.com/index.html." [Online]. Available: http://www.opnet.com/index.html

[25] "Omnet++ : http://www.omnetpp.org/." [Online]. Available: http://www.omnetpp.org/

[26] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols," in *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, ser. MobiCom '98. New York, NY, USA: ACM, 1998, pp. 85–97. [Online]. Available: http://doi.acm.org/10.1145/288235.288256

[27] N. Meghanathan, "On the connectivity, lifetime and hop count of routes determined using the city section and manhattan mobility models for vehicular ad hoc networks," in *Contemporary Computing*, ser. Communications in Computer and Information Science, S. Ranka, S. Aluru, R. Buyya, Y.-C. Chung, S. Dua, A. Grama, S. K. S. Gupta, R. Kumar, and V. V. Phoha, Eds. Springer Berlin Heidelberg, 2009, vol. 40, pp. 170–181.

[28] H. Arbabi and M. Weigle, "Highway mobility and vehicular ad-hoc networks in ns-3," in *Simulation Conference (WSC), Proceedings of the 2010 Winter*, dec. 2010, pp. 2991 –3003.

[29] A. Mahajan, N. Potnis, K. Gopalan, and A.-I. A. Wang, "Urban mobility models for vanets," in *IN PROC. OF 2ND WORKSHOP ON NEXT GENERATION WIRELESS NETWORKS*, 2006.

[30] X. Hong, M. Gerla, G. Pei, and C.-C. Chiang, "A group mobility model for ad hoc wireless networks," in *Proceedings of the 2nd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, ser. MSWiM '99. New York, NY, USA: ACM, 1999, pp. 53–60. [Online]. Available: http://doi.acm.org/10.1145/313237.313248

[31] M. Piórkowski, M. Raya, A. L. Lugo, P. Papadimitratos, M. Grossglauser, and J.-P. Hubaux, "Trans: realistic joint traffic and network simulator for vanets," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 12, no. 1, pp. 31–33, Jan. 2008. [Online]. Available: http://doi.acm.org/10.1145/1374512.1374522

[32] J. Härri, F. Filali, C. Bonnet, and M. Fiore, "Vanetmobisim: generating realistic mobility patterns for vanets," in *Proceedings of the 3rd international workshop on Vehicular ad hoc networks*, ser. VANET '06. New York, NY, USA: ACM, 2006, pp. 96–97. [Online]. Available: http://doi.acm.org/10.1145/1161064.1161084

[33] "Vissim." [Online]. Available: http://www.vissim.de/index.php?id=1801

[34] "Straw." [Online]. Available: http://www.aqualab.cs.northwestern.edu/projects/STRAW/index.php

[35] N. Surayati and M. Usop, "Performance evaluation of aodv , dsdv and dsr routing protocol in grid environment," *Journal of Computer Science*, vol. 9, no. 7, pp. 261–268, 2009. [Online]. Available: http://paper.ijcsns.org/07_book/200907/20090737.pdf

[36] P.-C. Cheng, K. C. Lee, M. Gerla, and J. Härri, "Geodtn+nav: Geographic dtn routing with navigator prediction for urban vehicular environments,"

Mob. Netw. Appl., vol. 15, no. 1, pp. 61–82, Feb. 2010. [Online]. Available: http://dx.doi.org/10.1007/s11036-009-0181-6

[37] C. Lochert, M. Mauve, H. Füssler, and H. Hartenstein, "Geographic routing in city scenarios," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 9, no. 1, pp. 69–72, Jan. 2005. [Online]. Available: http://doi.acm.org/10.1145/1055959.1055970

[38] M. Jerbi, S.-M. Senouci, T. Rasheed, and Y. Ghamri-Doudane, "Towards efficient geographic routing in urban vehicular networks," *Vehicular Technology, IEEE Transactions on*, vol. 58, no. 9, pp. 5048 –5059, nov. 2009.

[39] J. Zhao and G. Cao, "Vadd: Vehicle-assisted data delivery in vehicular ad hoc networks," in *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, april 2006, pp. 1 –12.

[40] I. Leontiadis and C. Mascolo, "Geopps: Geographical opportunistic routing for vehicular networks," in *World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007. IEEE International Symposium on a*, june 2007, pp. 1 –6.

[41] C. Schwingenschlogl and T. Kosch, "Geocast enhancements of aodv for vehicular networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 6, no. 3, pp. 96–97, Jun. 2002. [Online]. Available: http://doi.acm.org/10.1145/581291.581307

[42] P. Ruiz, B. Dorronsoro, P. Bouvry, and L. J. Tardon, "Information dissemination in vanets based upon a tree topology," *Journal of Ad hoc Networks*, vol. 10, no. 1, pp. 111–127, 2012.

[43] N. Wisitpongphan, O. Tonguz, J. Parikh, P. Mudalige, F. Bai, and V. Sadekar, "Broadcast storm mitigation techniques in vehicular ad hoc networks," *Wireless Communications, IEEE*, vol. 14, no. 6, pp. 84 –94, december 2007.

[44] T. Nadeem, P. Shankar, and L. Iftode, "A comparative study of data dissemination models for vanets," in *Mobile and Ubiquitous Systems: Networking Services, 2006 Third Annual International Conference on*, july 2006, pp. 1 –10.

[45] T. Zhao, Z. Liu, W. Yan, and X. Li, "Bfbd: A bloom filter based buffering data dissemination algorithm for vehicular ad hoc networks," in *Consumer Communications and Networking Conference (CCNC), 2011 IEEE*, jan. 2011, pp. 447 –481.

[46] G. Samara, W. Al-Salihy, and R. Sures, "Efficient certificate management in vanet," in *Future Computer and Communication (ICFCC), 2010 2nd International Conference on*, vol. 3, may 2010, pp. V3–750 –V3–754.

Glossary

List of Acronyms

VANET	Vehicular Area Network
V2V	Vehicular to Vehicular Communication
RSU	Road Side Unit
CPU	Central Processing Unit
POI	Point of Interest
OEM	Original equipment manufacturer
PND	Portable Navigation Devices
ADAS	Advanced Driver Assistance Systems
ADB	Adaptive and Distributive Broadcasting algorithm
CAN	Controller Area Network
NDS	Navigation Data Standard