Dynamic Simulations of VANET Scenarios

Hassan Artail, Malak Safieddine, Tania Safar, Malek El-Khatib, Tarek Ibrahim, Kassem Fawaz

Department of Electrical and Computer Engineering
Faculty of Engineering and Architecture
American University of Beirut- Lebanon
E-mails: {hartail, mhs32, tms08, mde01, toi00, kmf04}@aub.edu.lb

***Abstract*- Vehicular ad hoc networks, also known as VANETs, constitute a major pillar in making the dream of an Intelligent Transportation System (ITS) come true. By enabling vehicles to communicate with each other, it would be possible to have safer and more efficient roads where drivers and concerned authorities are supplied with timely information. Based on a short to medium range communication systems, VANETs can enable both safety and entertainment types of applications to come to reality. Unfortunately, the application layer has not received sufficient attention. Although some of the undergoing projects have touched on the subject, their works do not seriously cover issues dealing with actual implementations of VANET scenarios. This paper describes some application layer scenarios which we developed using the network simulator ns2. We describe the limitations of ns2 as it concerns VANET simulations and our implemented solution, and then move on to considering car braking and changing lane scenarios in order to demonstrate how such applications may work.**

Keywords – VANETs; Safety applications; traffic simulations; ns2

I. Introduction

With the emergence of Mobile Ad hoc Networks (MANETs), researchers conceptualized the idea of communicating vehicles giving rise to Vehicular Ad-hoc Networks (VANETs). VANETs are the main focus of engineers yearning to transform cars into intelligent machines that communicate in order to make the journey as safe and as entertaining as possible. In trying to achieve this goal, several research projects have been launched, most of which focus on lower layers of the network architecture, especially the physical and data link layers. A whole standard, the Dedicated Short Range Communications Standard (DSRC) has been developed, devoting a short to medium frequency range to automotive applications. Unfortunately, the application layer and its implementation have not captured the needed attention. This constituted the motivation behind our work on this project to simulate real-life VANET scenarios using the well-known ns2 simulator.

VANETs applications can be divided into two major broad categories: Public safety applications and non-public safety or comfort applications. These applications range from the reduction of road causalities by means of brake warnings, intersection assistance, and collision avoidance systems to offering guidance to available parking lots, discovering the traffic situation on a planned route, and coordinating car flow and traffic lights. This paper stresses on the safety applications which are more fundamental for building a functional Intelligent Transportation System (ITS).

The possibilities and opportunities of VANETs are limitless but so are the challenges associated with their implementations. Although VANETs represent a special case of MANETs, there are major differences between the two. To begin with, compared to MANETs, the topology of VANETs varies at a higher rate and thus suffers from frequent network fragmentation. It is important to note that with the initial deployment of VANETs, only a limited number of vehicles will be equipped with transceivers, which might worsen the problem of fragmentation. Several solutions for such problems have been proposed like increasing the transmission power and exchanging routing information in tables, however, they also suffer from many limitations concerning bandwidth, throughput, and cost. Another major challenge associated with VANETs is their vulnerability to several privacy and security breaches due to the large number of independent network members and the presence of the human factor. Therefore, certain measures must be imposed on vehicle-to-vehicle communication in order to alleviate security concerns.

In the following section we discuss previous work that is available in the literature and concerns vehicle ad hoc networks. We then move on to describing the simulation tool we used before presenting the details of our simulations. Finally, we conclude with findings and recommendations.

II. Related Work

To the best of our knowledge, no inter-vehicular communication systems for data exchange among vehicles or between vehicles and roadside units have been put into operation. Nevertheless, the advancements in wireless technologies and the rising awareness about the importance of safety measures in transportation systems have motivated research groups and companies to focus on solving fundamental and infrastructural VANET problems. This research was mostly oriented towards VANET simulation. As a matter of fact, there exist in the literature various simulation schemes that could be employed, but surprisingly, very little work has been done to actually simulate realistic VANET scenarios.

In [1] the simulation problem in VANETS is divided in to two parts, the first is to develop sound mobility models, both at the macroscopic and microscopic levels, while the second one is concerned with combining the developed mobility model with the used network simulator. Today there is a range of existing mobility generators that provide compatibility with several network simulators like ns2, GlomoSim, Qualnet, SWANS, and others. In this respect we mention a few, like *VanetMobiSim* [8], *STRAW* [4], *SUMO* [13], *Virtual Track* [16], and *AutoMesh* [14].

A major issue that concerns these simulation tools is that the traffic generators and network simulators do not typically interact at run time, mostly because the various network simulators existed long before the traffic mobility generators, and were not designed with VANET applications in mind. The authors in [12] presented an integrated tool that combines two mobility models (IDM and MOBIL) with the network simulator OMNET to assess the performance of network layer protocols, while focusing on a specific routing scheme, namely the DYnamic MANET On demand (DYMO) protocol. It was shown that the proposed mode of coupling improves the accuracy of network layer performance characterization. Very recently, the authors in [15] proposed a system approach named *Traffic Control Interface* (*TraCI*) that is based on live interactions between the traffic generator and the network simulator using TCP/IP socket communication between each simulated node (vehicle) and the traffic generator. Actually, the authors provide an implementation for the interaction between the traffic generator SUMO and the network simulator ns2. In this paper, and in the context of dynamic VANET simulations, we took a different and more efficient approach. Instead of having the traffic generator communicate with every node, we only used it to a priori produce the mobility file (as is usually done), and then through socket communication and multithreading we allow client applications to interact with the scheduler of the simulator (ns2) to alter already scheduled events or to add new ones. Such applications could represent the driver or event handlers that react to dynamic road or weather conditions.

In the following we briefly discuss two projects that have been initiated by consortiums which include automotive companies, technology suppliers, and academic institutions.

A. *FleetNet – Internet on the road*

The FleetNet project started in Germany in September 2000 and was set up by a consortium of six companies and three universities including *DaimlerChrysler AG*, *Robert Bosch Gmbh* and *Technische Universität Hamburg*. As outlined in Enkelmann's paper [6], the main objective of FleetNet was to develop a platform for inter-vehicular communication systems achieved by examining mobile ad hoc radio networks, through three classes: First, cooperative driver-assistance applications for safety; Second, local floating car data applications for information updates; and Finally, user communication and information services for entertainment [6]. Although this project aims to handle the various layers of the network architecture, the project's website does not provide information related to the progress of the work nor about the implementation process. Moreover, the simulations being conducted are not apparently disclosed to the public, which makes it hard to build upon any achieved developments.

B. *CarTALK 2000*

CarTALK 2000 was started in August 2001 and funded within the ITS Cluster of the Fifth Framework Program of the European Commission. With the coordination of *DaimlerChrysler*, the project consortium assimilates the knowledge of European car manufacturers, IT industry, suppliers, and some expert research institutes. Its main objectives are the development of cooperative driver assistance systems as well as self-organizing ad-hoc radio networks [2]. CarTALK identifies three applications types: Information and Warning Functions, communication-based longitudinal control, and cooperative assistance systems. The last category comprises applications that focus on exchanging information among vehicles to assist the driver in resolving critical situations [2].

Other comparable projects also exist. Most notably, we mention the Simulation of Car-to-Car messaging project [5], the Network on Wheels (NOW) project [10], the CAR-2-CAR Communication Consortium [3], and the CarLink project. The latter project developed in the university of Malaga aims to provide users with live data concerning weather and traffic through vehicle-to-vehicle and vehicle-to-infrastructure communication. The simulation in Carlink is done mainly using VanetMobiSim as described in [9].

One can clearly see that a considerable number of projects have already been launched with the majority concentrating on exploring the physical and network layers of inter-vehicle communication. The application layer, however, remains theoretical where only suggestions of possible applications are provided. The aforementioned projects might have achieved progress in implementing this layer, but such progress remains mostly unpublished, possibly leading to duplication of work.

III. Network Simulator (ns2)

Simulation is the dynamic representation of a real life problem achieved by building computer models and running them under sensible assumptions. Since in our project validating the applications on a real vehicular network is infeasible, we used a simulation tool for analytically solving, testing, evaluating and demonstrating our proposed course of action. The network simulator ns2 [11], which we will discuss below, was employed to define the constraints and reactions of the vehicles being driven on the road.

NS2 is a discrete event simulator that models various network protocols including wired, wireless, TCP, UDP, ad hoc routing, infrastructure, etc. In our project, we used it to simulate real-time vehicle driving scenarios. The simulator is composed of four major components which are the ns simulator itself, the network animator or NAM which is responsible of visualizing various outputs and providing the interface to generate ns scripts, the pre-processing subsystem that handles traffic and topology generation, and the post-processing subsystem which performs simple trace analysis.

A simulator has two basic tasks and therefore uses two different programming languages. On one hand, it simulates detailed protocols and efficiently manipulates bytes and packet headers, and implements algorithms that run over large data sets. On the other hand, it explores a number of scenarios through varying parameters and configurations. For satisfying the first job, it needs a programming language which is very fast to run but slower to change, namely C++. As for the second, OTcl is applicable because it runs much slower but can be changed very quickly [7]. The simulator employs tclcl binding to provide the glue, which makes objects and variables appear to both languages.

Regarding functionality, ns2 is capable of covering the basic issues like setting up nodes and links, sending data from one node to another, monitoring a queue and starting NAM from tcl scripts to visualize the simulations. Moreover, ns2 provides features for running wireless simulations where one can define mobile nodes for which various network component parameters can be specified, including the link layer type, MAC protocol, the antenna type, the radio-propagation model, and the wireless channel. The mobility models of the nodes can be either obtained from traffic-pattern files available with the ns distribution, or by inputting trace files generated by vehicle traffic simulation programs.

A. *Limitations*

The network simulator ns2 employs a single-threaded event-driven scheduler. The simulator has two basic event types, namely "at-events" and "packets". Each comprises a unique ID, a firing time, and a handler function to execute. At the beginning of the simulation the TCL file is read and evaluated so that all events are generated and inserted into the appropriate data structure. The scheduler maintains an internal clock in seconds to determine the execution times of events. Events are executed one at a time, and an event will not start execution until the one immediately before it in the scheduler's data structure is fully processed.

Once the simulation is started, the user or a runtime process cannot intervene to change the timings of events or add new ones. In VANET simulations this restriction can lead to unrealistic scenarios. For example, changing lane events and braking episodes depend on real-time traffic dynamics whose types and timings may not be known a priori, and thus cannot be specified in the TCL code at code-writing time. Hence, there is need for integrating into ns2 capability for dynamic simulations in order to implement realistic VANET scenarios.

B. *Implemented Solution*

To add dynamic simulation capability into ns2, multithreading functionality had to be implemented into the scheduler (actually in the constructor of the scheduler, so as to create the thread upon instantiation). A handler function was introduced which runs in an infinite loop waiting for input from a user application through a TCP socket. Theoretically, this input should consist of a TCL statement to be executed. The TCL statements are fed to the newly created thread through a client application which opens a TCP connection with the scheduler. Simulated cars are defined in the TCL code as an array of nodes, which makes the communication between the client (user application) and the server (scheduler) straightforward. Through monitoring the progress of the simulation (i.e., changing lanes, breaking events, travel speed, etc.) in real time, the user application can react by inserting events concerning particular cars with handlers to be executed at specified times. Finally, we note that using multithreading allows for employing third party agents that can act as drivers which feed braking and changing lane events to the simulation at runtime, and therefore turn the simulator into an emulator.

IV. Implementation

This section presents the three different scenarios that we implemented with descriptions and simulation snapshots.

A. *The Braking Scenario*

The breaking scenario simulates the topology's reaction when a driver of a vehicle suddenly pushes the brakes. It respects the following general logic: the braking vehicle broadcasts a braking packet to inform its surrounding, only concerned vehicles will decelerate according to their relative positions. In what follows, we explain the details of this mechanism and show snapshots of the resulting NAM visualizations.

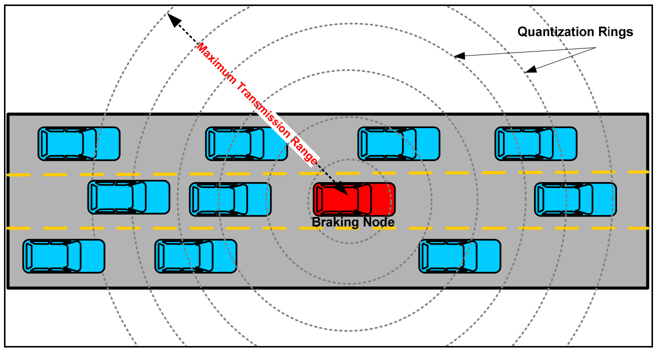


Fig. 1. Braking Scenario Diagram

1) Technical Description

The main steps for simulating this scenario include attaching to each node in the topology the Brake Agent, which maintains its status and controls its behavior in braking situations. Besides, the Brake Packet Header, of size 64B, is fabricated to include the main information needed in the inter-node communication. Finally, the TCL script reads the nodes movements file and initiates, under certain conditions, the C++ agent functions.

In our simulation, we use in the TCL script a command specific to our Brake agent that invokes a random node to stop at a certain time instant. As a result, this node will periodically broadcast, over a defined transmission range

(T_{max}), a Brake packet containing its current position that includes the road and lane identifiers. As shown in figure 1, we quantized the covered area into R co-centric rings. Among the nodes within this range, only those in the same road and lane behind the stopping car will respond to the alarm. Each will compute the distance (*d*) separating it from the source and locate the ring (*r*) it belongs to. The calculated ring constitutes the base to compute the new velocity (V_{new}), which is inversely proportional to the distance. Thus, as the vehicle gets closer to the source, it slows down until it completely stops before hitting the vehicle just in front it, and forwards the packet to its neighbors. The ring and speed are computed as follows:

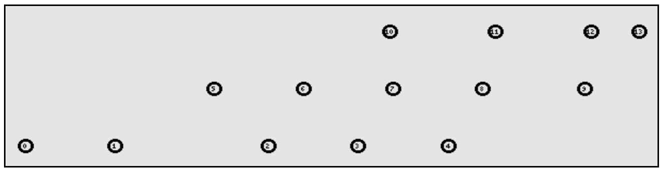
$$r = \frac{d}{T_{\max}} * R + 1 \qquad V_{new} = V_{old}\left(1 - \frac{1}{P}\right)$$

Moreover, nodes behind the stopping node and in different lanes will refrain from changing to that node's lane to avoid congestion. This receiving node behavior is illustrated in the flowchart of Figure 2.

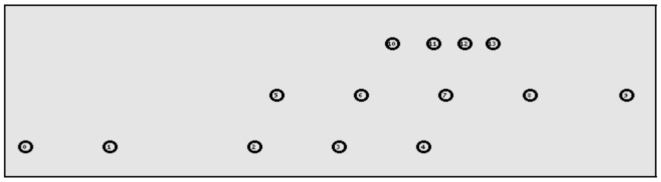
2) Simulation Snapshots

To validate our theoretical implementation, we visualize the topology using NAM. Below are successive NAM snapshots for a braking simulation, with the first node (car) of the upper lane stopping. Note how the nodes behind it are shown stopped behind each other, and how the nodes in the lower two lanes kept on going and passed by the stopped nodes. Also note that the nodes of the bottom lane are moving at a lower speed relative to the nodes in the middle lane.

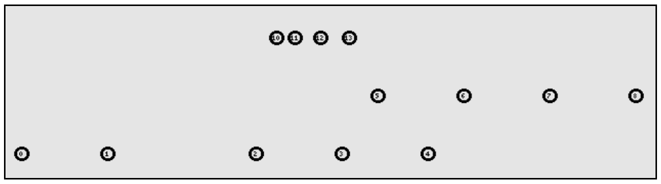
Simulation time = 9.37sec:



Simulation time = 12.43sec:



Simulation time = 16.47sec:



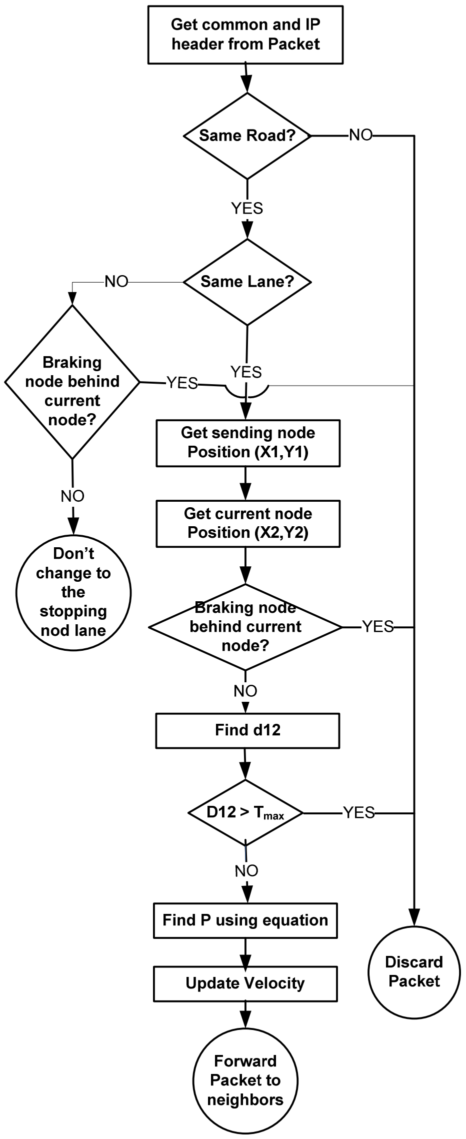

Get common and IP header from Packet
Same Road?
NO
YES
Same Lane?
NO
YES
Braking node behind current node?
YES
NO
Don't change to the stopping nod lane
Get sending node Position (X1,Y1)
Get current node Position (X2,Y2)
Braking node behind current node?
YES
NO
Find d12
D12 > Tmax
YES
NO
Find P using equation
Update Velocity
Forward Packet to neighbors
Discard Packet

Fig. 2. Flow Chart for the Braking Algorithm

B. *Changing Lanes Scenario*

This second safety scenario covers the case of a car changing its lane and the applicable safety rules regulating its behavior and that of its neighbors in its destination lane.

1) Technical Description

Again, we developed a Change Lane packet header and a Change Lane agent attached to all nodes. Through the TCL script, we trigger a random vehicle to move to a certain destination lane at some point of time. As s result, the vehicle broadcasts over a certain range a Discovery packet containing its position, speed, road, lane and destination lane, along with a timestamp common to all packets related to this incident. The purpose of this packet is to examine the safety of the road at that instance of time. Any node within the range receiving this Discovery packet will check if it is addressed to it by verifying that it is traveling in the assigned destination lane. Each related node will figure out its relative distance to the source. Here, we define a range in the destination lane, in front and behind the original vehicle, as illustrated in figure 3, in which the presence of any other car could lead to an accident. Accordingly, we might have three different situations:

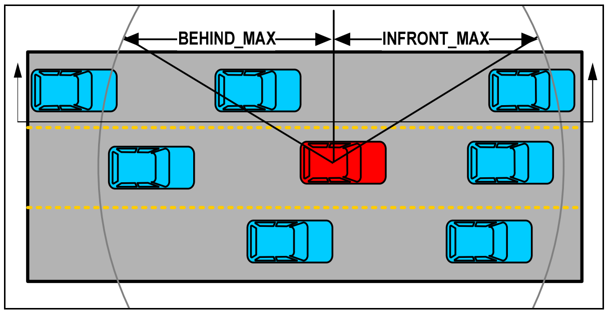


Fig. 3. Relative positions of the car changing lanes and its neighbors

i. If the receiving node falls in this range, it prepares a Negative Acknowledgement packet and sends it to the original node asking it not to cross.
ii. If it is outside this range in front of the concerned car, it means that it will not endager the crossing, and thus it replies with a Positive Acknowledgement, thus giving it the green light to cross.
iii. If the receiving node is behind the range, there is still a probability for collision with the crossing car, depending of the speeds of the two cars. The receiving node caclculates the time it takes the notifying car to reach its stable position on the new lane (assuming the node will move in a 45^o direction) and evaluates the time needed, using its current position and velocity, to reach this same point. Based on these time figures, if the cars intersect, the receiving node responds with a Negative Acknowledgement packet, otherwise with a Positive Acknowledgement packet.

After analyzing the different possible responses received from the neighboring nodes, the notifying node (i.e., the one that intends to change lanes) proceeds as follows. Before describing this however, it is important to note that the Change Lane agent at the notifying node keeps track of whether any response was received in response to the Discovery packet that was sent. In addition, it monitors for any Negative Acknowledgement that prohibits the car from changing its lane. The following is a summary of the behavior of the notifying node.

- If during a certain time interval, no positive or negative acknowledgement packets were received, it is implied that the destination lane is empty and that it is safe for the car to cross. In technical terms, if the handler times out while the agent still indicates that no reply was received, it invokes the function for changing lanes.

- If a Negative Acknowledgement packet is received, and corresponds to the same time stamp of the Discovery packet that the original node has broadcasted, no other positive or negative packets will be taken into consideration any more. Afterwards, using another handler, the node will wait for some time and repeat the whole process of examining the conditions of the road using a new Discovery packet.

- If a Positive Acknowledgement packet is received, before a negative acknowledgement, it will be understood that the dangerous zone defined earlier is empty. This is because the distance between the notifying node and any other node in this region is smaller than the distance between it and a node outside the range, and thus Negative Acknowledgement packets from inside the region will be received before Positive Acknowledgement packets from outside it. However, this does not mean that it is completely safe now for the car to cross. This is due to the simple reason that additional Negative Acknowledgement packets might be sent from cars behind the zone but with a velocity that indicates a possibility for collision as explained before. Therefore, in our code we differentiated between Positive Acknowledgement packets arriving from in front of the range, and those arriving from behind it. If the packet is received from the front, a handler is called for the node to wait for a short time interval before crossing, to be on the safe side and to account for any negative acknowledgements that are on the way. On the other hand, if the Positive Acknowledgement packet is received from behind, the car directly calls the function of changing lane and moves to its destination lane.

After discussing the different cases we noe illustrate how the node will change its lane when the time is appropriate. First, we assume that the car changes its lane while maintaining the same velocity it had while travelling on the original lane, follws a $45^{\mathbf{o}}$ path, and continues on the new lane with a velocity appropriate to this lane (i.e., same as that of those cars travelling on the new lane). In the changing lane function, the node calculates the time it needs to perform the change and saves it. Then, the *setdest* ns-2 command is called to move it to the other lane. After that and after having the handler function wait for the time just discussed, the *setdest* command is called again to make the node move to its initial destination set in the *movement file*, but along the modified path.

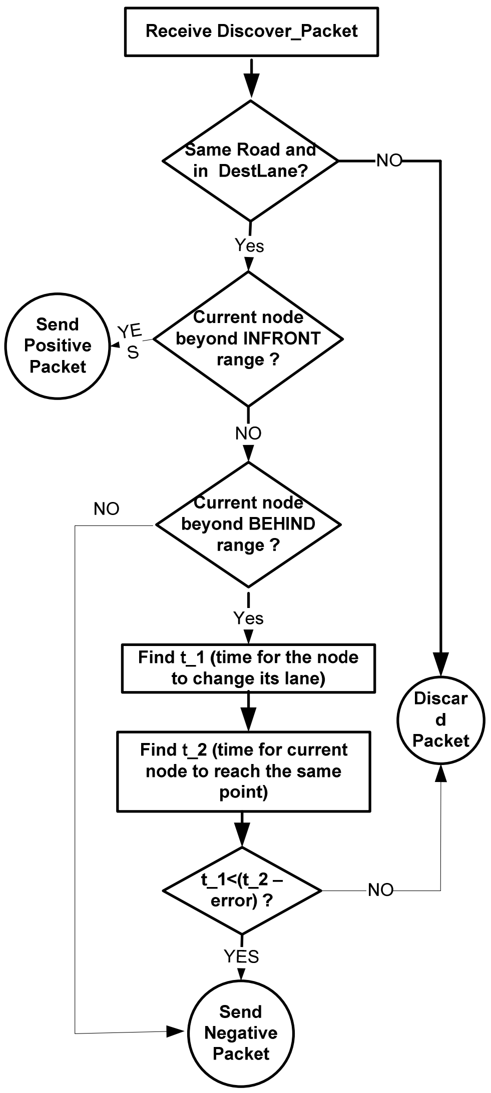

Receive Discover_Packet
Same Road and in DestLane?
NO
Yes
Current node beyond INFRONT range ?
YES
Send Positive Packet
NO
Current node beyond BEHIND range ?
NO
Yes
Find t_1 (time for the node to change its lane)
Find t_2 (time for current node to reach the same point)
t_1<(t_2 – error) ?
NO
YES
Discard Packet
Send Negative Packet

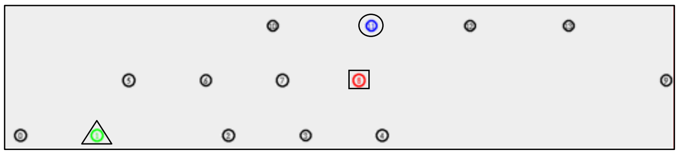
Fig. 4. Flow Chart of Receiving a Discovery request

The above is summarized in the charts of Figures 4 and 5.

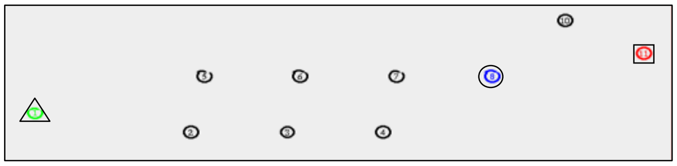
2) Simulation Snapshots

The following sequence of snapshots demonstrate the relative motion of the vehicules and their reactions when the red (square), blue (circle), and green (triangle) nodes simultaneously decide to move to left, right, and left lanes respectively, at time = 5sec. Note that the left lane is the fastest relative to the other two ones.

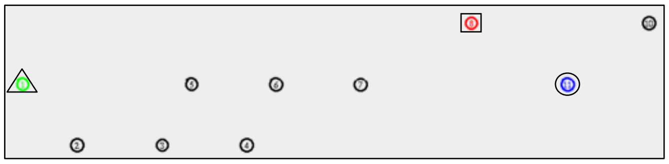
Simulation time = 5.26sec:



Simulation time = 8.62sec.



Simulation time = 11.5 sec:.



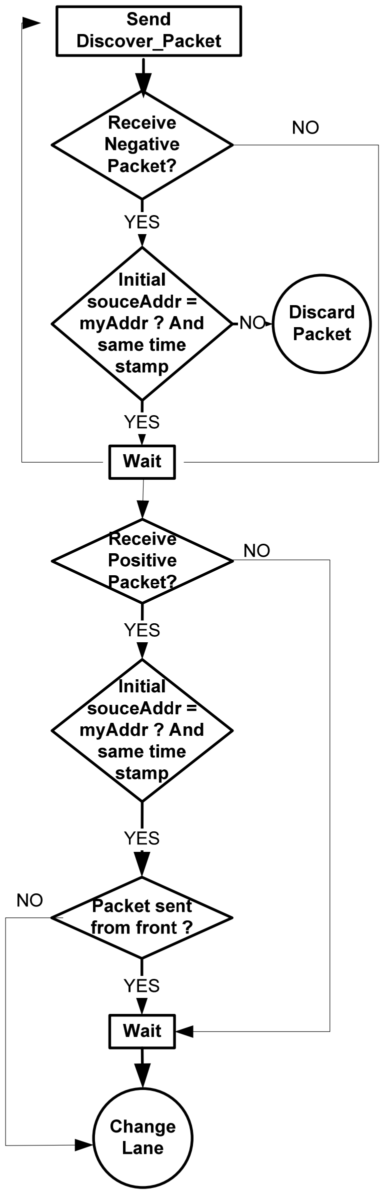


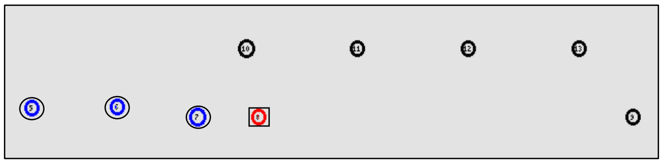
Fig. 5. Flow Chart of Changing Lanes

C. Braking with Changing Lanes Combined

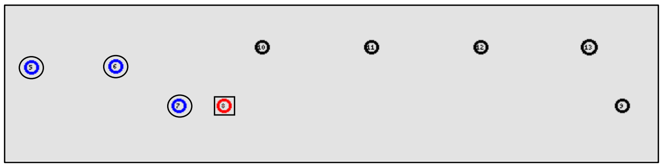
In the two previous scenarios, we implemented the braking and changing lanes algorithms as separate agents, where the nodes in the simulation were equipped with one or the other. In this scenario, we will combine the two algorithms in order to create a new agent that is capable of performing both of the two operations based on the situation encountered within the movement file. During the simulation, one or more nodes may attempt to brake while other nodes that are affected by this braking (moving nodes behind the braking ones) will try to change lanes in order to overcome the obstacle and reach their final destinations. We will not thoroughly describe the implementation in this section since it is a mere combination of the ones presented in the earlier sections with minor modifications.

The following sequence of snapshots demonstrate the motion of the vehicules in blue (circles) and their reaction when the red (square) node decides to brake at 6sec.

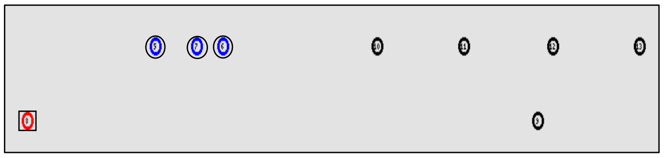
Simulation time = 6.43sec:



Simulation time = 6.84sec:



Simulation time = 10.02sec:



V. Conclusion

In this paper, we examined Vehicular Ad Hoc Networks and studied their main challenges and potential applications. The presence of such vehicular systems, if implemented in the future, will create opportunities for developing and deploying safety and leisure applications between vehicles. However, this versatility does not come for free, as there are numerous challenges that will be faced due to communication network flooding and high security risks.

Believing that the work on such complex topic is a cumulative effort and requires an accumulation of research and work, we decided to be part of this new global research and add some contribution to it. First, we added a modification to the ns2 simulator in order to enable it to realistically simulate dynamic VANET scenarios. We simulated three real life scenarios of vehicles on the road. With the braking algorithm, we made sure that nodes behind the braking car decrease their speed gradually and gracefully. The second scenario, which concerns changing lane scenarios aimed at ensuring safe lane changes by the cars on the highway in order to minimize and even eliminate potential accidents. The third and final scenario was the combination of the previous two scenarios and intended to simulate more realistic traffic conditions, in which events would happen successively and wise decisions would be taken. It is worth noting that convenience-related applications can be added in the future and security threats can be accounted for without impacting the current implementation significantly.

Finally, we acknowledge that in the descriptions of the implemented scenarios we may have implied that the car will react to the various conditions by taking braking and lane-change actions autonomously. We note that this was done to simplify the discussion. In real-life, the onboard system in the car will have to engage the driver through warnings and sometimes using alarms when the situation worsens (i.e., when an action on the driver's part is deemed by the system to be causing a potential accident).

References

[1] C. Bonnet, J. Harri, and F. Filali, "Mobility Models for Vehicular Ad Hoc Networks: A Survey and Taxonomy", Research Report RR-06-168, Institut Eurécom, March 2007.

[2] CarTALK2000 project, August 2001. Available http: www.cartalk2000.net/

[3] Car-to-Car Communication consortium, http://www.car-to-car.org

[4] D. Choffnes, F. Bustamante, "An Integrated Mobility and Traffic Model for Vehicular Wireless Networks", 2nd ACM international Workshop on Vehicular Ad Hoc Networks (VANET 2005), September 2005, Cologne, Germany.

[5] S. Eichler, T. Kosch, B. Ostermaier, and C. Schroth, "Simulation of Car-to-Car Messaging: Analyzing the impact on Road Traffic". *13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, September 2005, Atlanta, GA.

[6] W. Enkelmann, FleetNet - applications for inter-vehicle communication, *IEEE IV2003 Intelligent Vehicles Symposium*, June 2003, Columbus, OH.

[7] K. Fall, K. Varadhan (Editors), ns Manual, URL: http://www.isi.edu/nsnam/ns/doc/index.html

[8] M. Fiore, J. Harri, F. Filali, C. Bonnet, "Vehicular Mobility Simulation for VANETs," *40th Annual Simulation Symposium*, March 2007, Norfolk, VA.

[9] J. Härri, M. Fiore, F. Fethi, and C. Bonnet, "VanetMobiSim: generating realistic mobility patterns for VANETs, in Proc. *3rd ACM International Workshop on Vehicular Ad Hoc Networks (VANET'06)*, September 29, 2006, Los Angeles, CA.

[10] Network On wheels project, 2004, http://www.network-on-wheels.de/

[11] NS2 simulator, http://www.insi.edu/nsnam/ns

[12] C. Sommer, I. Dietrich, F. Dressler, "Realistic Simulation of Network Protocols in VANET Scenarios," 2007 Mobile Networking for Vehicular Environments (MOVE 07), May 2007, Anchorage, AK.
[13] SUMO - Simulation of Urban MObility, URL: http://sumo.sourceforge.net
[14] R. Vuyyuru, K. Oguchi, "Vehicle-to-Vehicle Ad Hoc Communication Protocol Evaluation using Simulation Framework", in Proc. Conference on Wireless on Demand Network Systems and Services, January 2007, Oberguyrgl, Austria.
[15] A. Wegener, M. Piorkowski, M. Raya, H. Hellbruck, S. Fischer, J. Hubaux, "TraCI: An Interface for Coupling Road Traffic and Network Simulators," 11th Communications and Networking Simulation Symposium (CNS 2008), April 2008, Ottawa, Canada.
[16] B. Zhou, K. Xu, M. Gerla "Group and Swarm Mobility Models for Ad Hoc Network Scenarios Using Virtual Tracks", in Proc. 2004 IEEE Military Communications Conference (MILCOM 04), October 2004, Monterey, CA.