Methods to Resolve Traffic Jams using VANET

Rohit Kumar
Department of Computer Sc. & Engineering
Chandigarh University, Gharuan
Mohali, Punjab

Abstract - In this paper we have proposed a method to avoid traffic jams when a vehicle met with an accident. A list of vehicles over a network is maintained. The basic idea is to find the shortest path whenever congestion is found or accident is detected or light goes red. The communication through a vehicle which met with an accident is blocked the algorithm will allow to find a safe path to continue to their final destinations without facing traffic jams. The main advantages of this algorithm are its simplicity and speed to find an alternate path whenever any obstacle comes into the way.

I. INTRODUCTION

We are representing automatic route finding in case of a Vehicular Adhoc Network. In this we use vehicles as a moving node. We have used a concept of traffic light, traffic, congestion and an accident in a scenario where we are considering a city area. In this scenario, the vehicle will perform an automatic route finding by taking the observation of surrounding traffic, external interference and the traffic light. The vehicle will search their path by using effective shortest path routing. In this network, we are taking a specific city area with one or two lane roads. At each junction there exists a traffic light which route the vehicles accordingly. As a vehicle detect that there is an accident or a congestion on a network then a message is build and sent to the base station through roadside units and to other vehicles coming on the same lane in same direction . After receiving the messages through other vehicles in same direction or through a base station, the vehicles will search their path automatically by taking the observation of surrounding traffic, obstacles on the way and the congestion on the path. The major problem is to find the appropriate route path to respective destinations and a network or geographical area. The connectivity problem too depends on these two factors.

II. EXPERIMENTAL STUDY

To evaluate the performance of VANET with a routing protocol, we have used VanetMobiSim-1.1 /NS-2 combination of application tool to run it. The result from Vanet MobiSim gives a realistic mobility model that supports both micro-mobility or macro-mobility features. Macro mobility model relates to road topology, road structure such as number of lanes, single way or double-way movement etc, traffic light constraints, speed limits where as micro mobility relates more to driving behavior. In present study, we have used a mobility model belonging to IDM-LC (Intelligent Driver Model with Lane changes) family. The output from Vanet MobiSim simulation is a traffic generator trace file that corresponds to position coordinates of each vehicular node at every time steps. This traffic generated trace file is the mobility model that goes through network simulation (by NS-2 package in present study) and ultimately generates a communication trace file (Fig.-1).

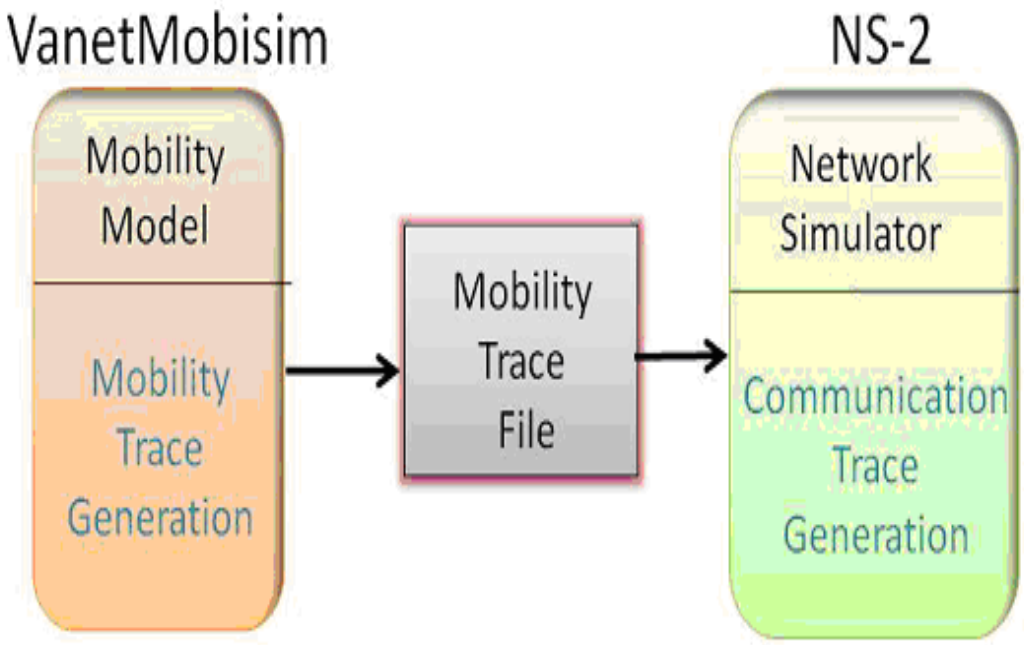

Figure 1: Simulation Architecture

The entire simulation and analysis workflow, we have carried out, can be divided into three distinct phases Workflow:

Phase-I	• Installation of Vanet MobiSim (in Windows) • Preparing ".xml" file for mobility model generator • Mobility Model trace file generation
Phase-II	• Installation of NS-2 (NS-allinone-2.34) on Linux Platform • Generating / editing 'OTCL' file for simulation • Using cbrgen.tcl file to include in primary '.tcl' file • Running simulation

	• Generating NS2 trace file and NAM
Phase-III	• Analyzing trace file to evaluate performance

Table 1: Phases of Simulation

III. PHASE-I: VANET MOBISIM AND MOBILITY MODEL

Vanet MobiSim is an extension to CANUMOBISIM (Communication in Ad Hoc Networks for Ubiquitous Computing for Mobility Model Simulation) [1] – a java based application with graphic user interface (GUI). Vanet MobiSim is an open source mobility generator model, specific to VANET scenario. It has capability to create realistic mobility model with high degree of realism. The application is compatible to both 'Window' or LINUX' platform and can be downloaded from the link " http://vanet.eurecom.fr/ " [24] and requires Java Run Time Environment version 1.5 or higher. The necessary guidance for successful installation is also provided in the link [2].

The output from Vanet MobiSim resulted in a mobility trace file (Fig.-1) which can be of any selected format compatible to NS-2, Qual Net, Glomo Sim or Op Net file for its further use to communication network simulation. For our present simulation, IDM-LC (with 20 nodes) NS-2 trace file is generated. Besides it can also dump a screen shot of the model defined in 'x fig' format.

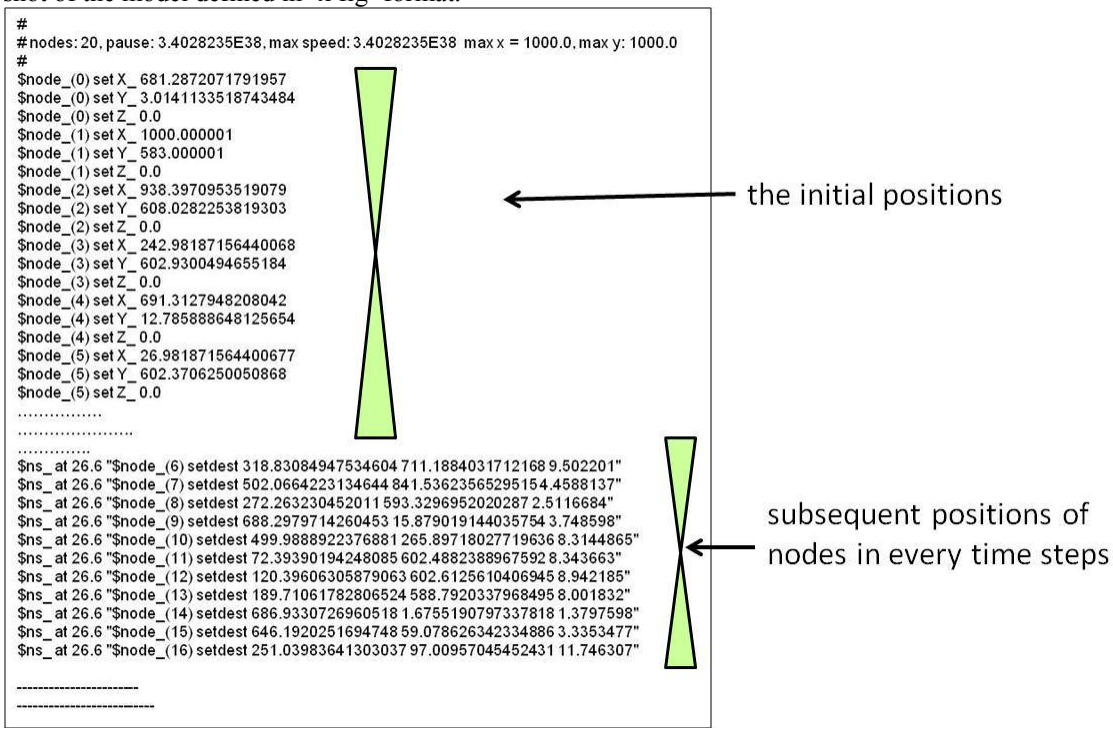

Figure 2: Example of mobility-trace file in NS-2 format

IV. PHASE-II: NS-2 FOR NETWORK SIMULATION

NS-2 is a discrete network simulator that provides significant simulation of transport, routing, and multicast over-wired and wireless networks. Ns-2 code is written using C++ and OTCL and is kept in a separate file that is executed by OTCL interpreter, thereby generating a communication trace file and a NAM (Network animator) file as its output. The output trace file describes the network topologies and log events that exhibit the output of the nodes communicating with each other and the NAM file animates the traces derived from the simulation and analyze

the events to understand the network behavior. In our case we have installed NS-2 (ns-allinone-2.34) in LINUX platform (Ibuntu-9.10) and a general procedure for this installation can be accessed from the web link [3, 4].
Once installed, we need to write (or edit the supplied one from the set of examples) the file in OTCL script language [5, 6] and execute NS with this file as an input. This OTCL file incorporates traffic generator trace file from VanetMobiSim and other communication parameters like routing protocol, CBR (constant bit rate) etc for using them in the simulation. A sketch of input/output flow-chart from NS-2 simulation is shown in Fig 3.

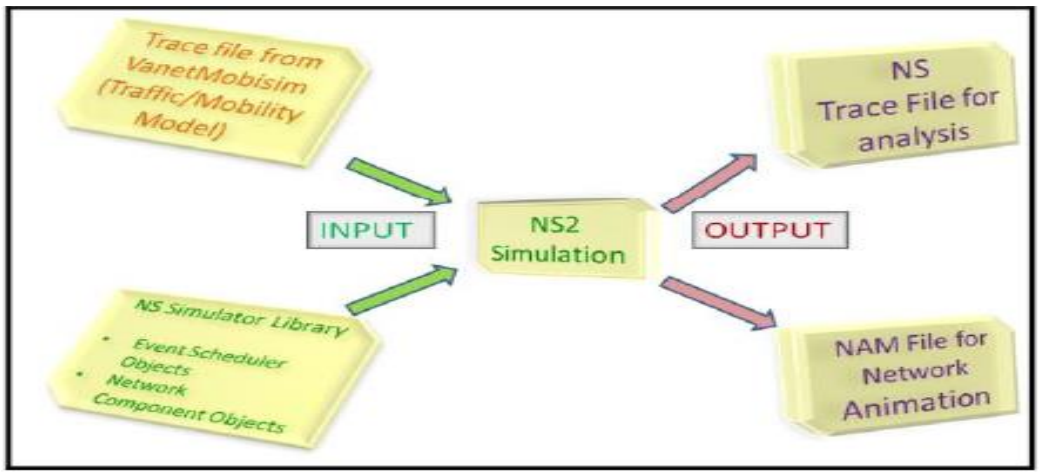

Figure 3: Input/output flow diagram of NS-2 Simulation.

As explained above (Fig.-3), the OTCL script for NS-2 simulation comprises inputs for event scheduler objects and network component objects including the nodes' mobility traces derived from Vanet MobiSim.
Phase-III: NS-2 Trace Analysis: Once we have generated NS-2 outputted trace file and network animator NAM file, the next phase of task lies in animating the NAM file to visualize node movement behavior and more importantly, extracting information from trace data file and appropriately plot and study them for performance evaluation.

V. NAM-FILE VISUALIZATION:

NAM (Network Animator) file generated [7] is mainly used to visualize topology and packet level simulation. It can also be used to show the routing path the packets took and how they are queued, transmitted or dropped. It can also monitor individual nodes, links or packets.

VI. IMPLEMENTATION

Scenario for Lane City Area

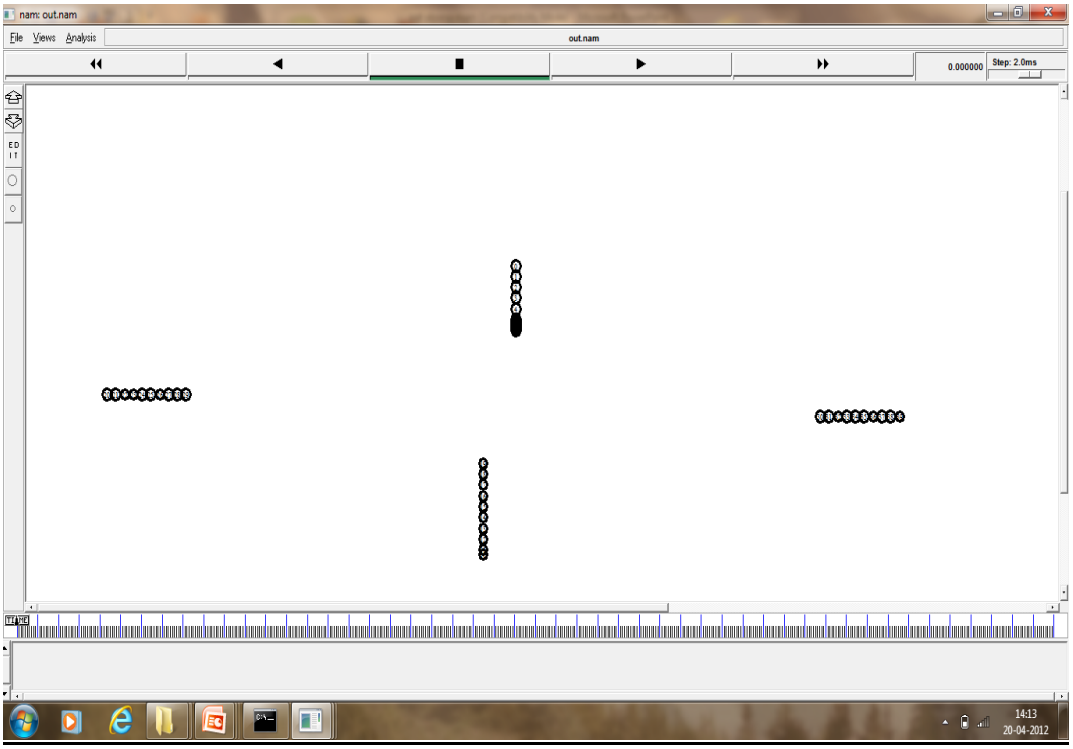

Figure 4: Lane City Area

In above snapshots, the vehicles are going on their way for the destination in opposite direction as that of the network partition. The vehicles are making queues at the intersection points. In these queues, the vehicles are at a particular distance apart from each other to avoid the collision. The waiting vehicles at intersection move towards their destination whenever they receive a green signal from the roadside infrastructure near to the traffic light signals.

VII. Scenario For Chances of Occurring Accident

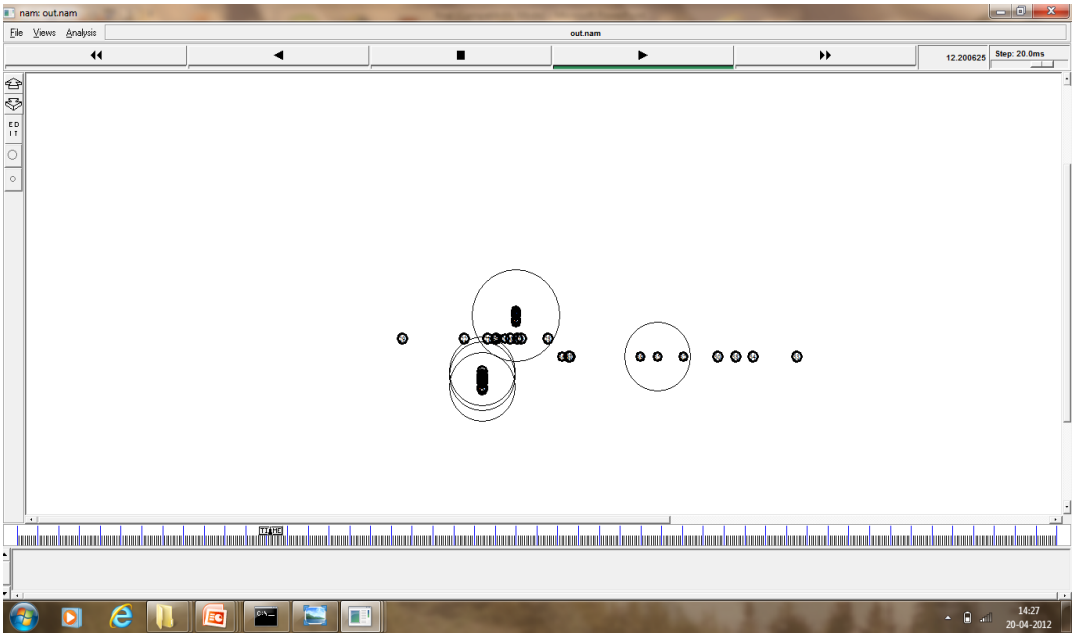

Figure 5: Chances of Occurring Accident

In figure, it shows that the vehicles are very close to each other according to specified lane. There are maximum chances of occurrence of the accident in such scenario as some user violates the traffic rules.

VIII. SCENARIO FOR ACCIDENT OCCURRED

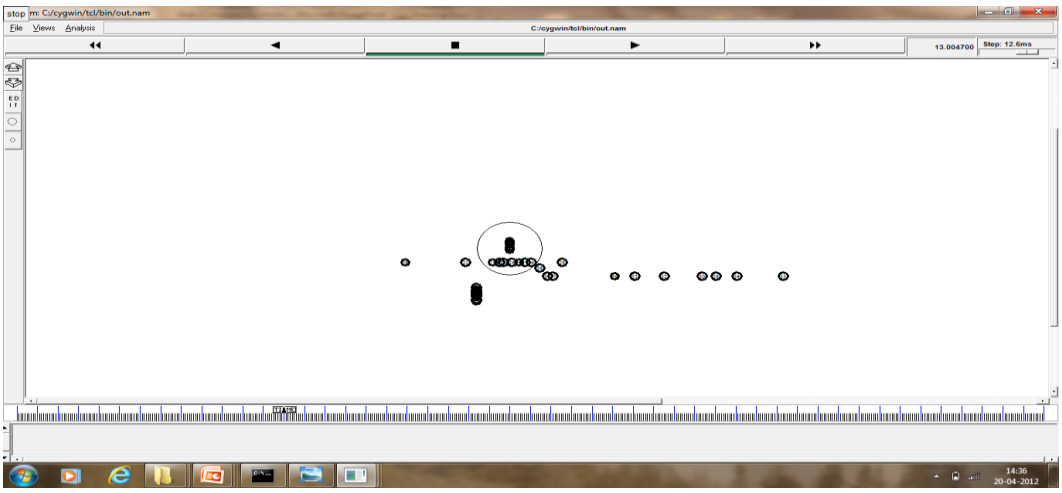

Figure 6: Accident Occurred

Here figure is showing the showing the situation where the accident has occurred. As the accident will occur the vehicle transmission in that area stop for a while and the vehicle start sending the messages to the coming vehicles about the accident.

IX. SCENARIO FOR TRANSMISSION PERFORMED

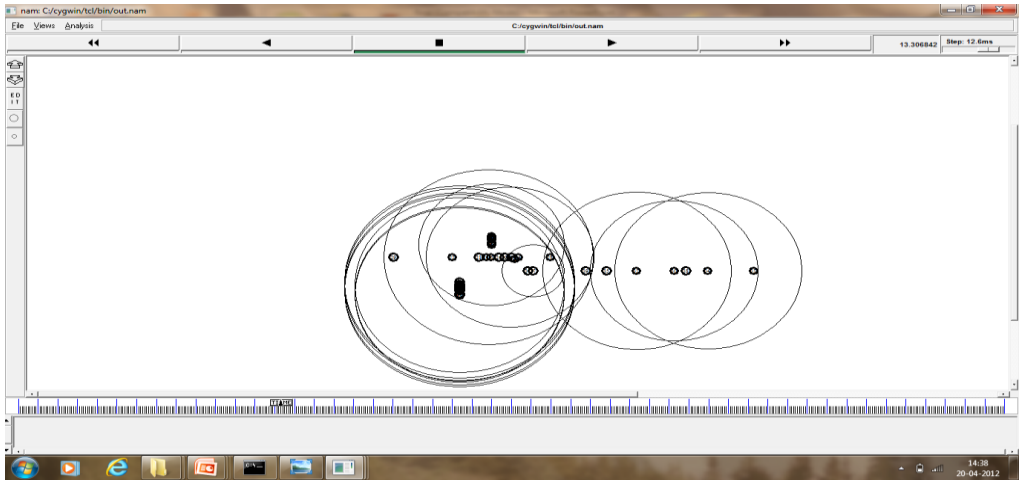

Figure 7: Transmission Performed

Figure 7 showing the vehicles that are not affected because of accident will continue the transmission without any interruption over the network.

X. SCENARIO FOR REFORMED TRANSMISSION

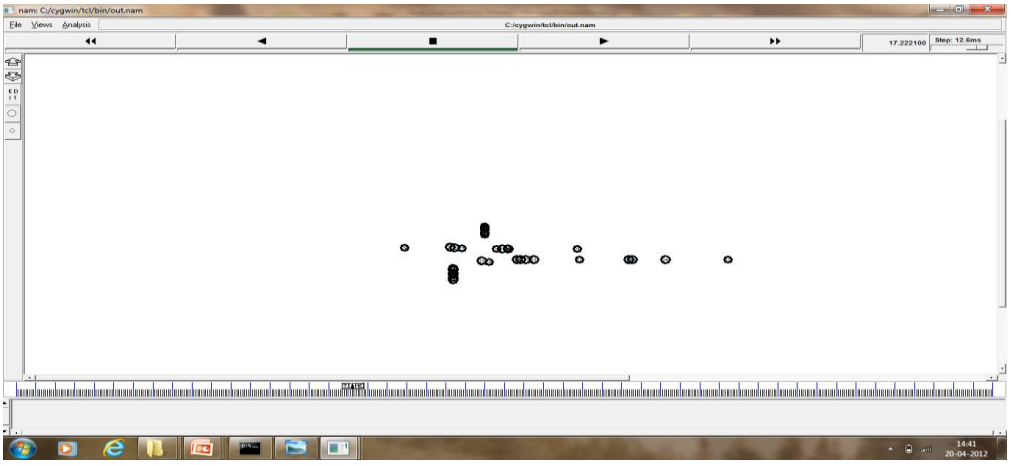

Figure 8: Reformed Transmission

Here figure 8 is showing the reformed transmission. As the accident occurs, the accident nodes inform all its neighboring nodes about his occurrence of the accident. As the vehicle get the information, other vehicle will change their route of communication over the network.

XI. ALGORITHM

Algorithm (Vehicle [],n, V)

/* vehicle is the list of n vehicle over the network, V is the current vehicle we are observing*/

{

1. for i=1 to n
2. {
3. if (Distance(Vehicle(i},V)< Threshold)
4. {
5. Maintain List(V,Info)=Vehicle(i)
6. }
7. }
8. if Accident(V)=True
9. {
10. Find the list of Accident Nodes
11. Block the communication for accident nodes
12. Mark the safe paths to keep on communication
13. For i=1 to length(List(V))
14. {
15. Message(ROUTECHANGE,Vehicle(i))
16. Vechile(i) will move on substitute route according to its requirement
17. }
18. }
19. For each vehicle perform communication effectively.
20. }

Figure 3.4 Proposed Algorithm

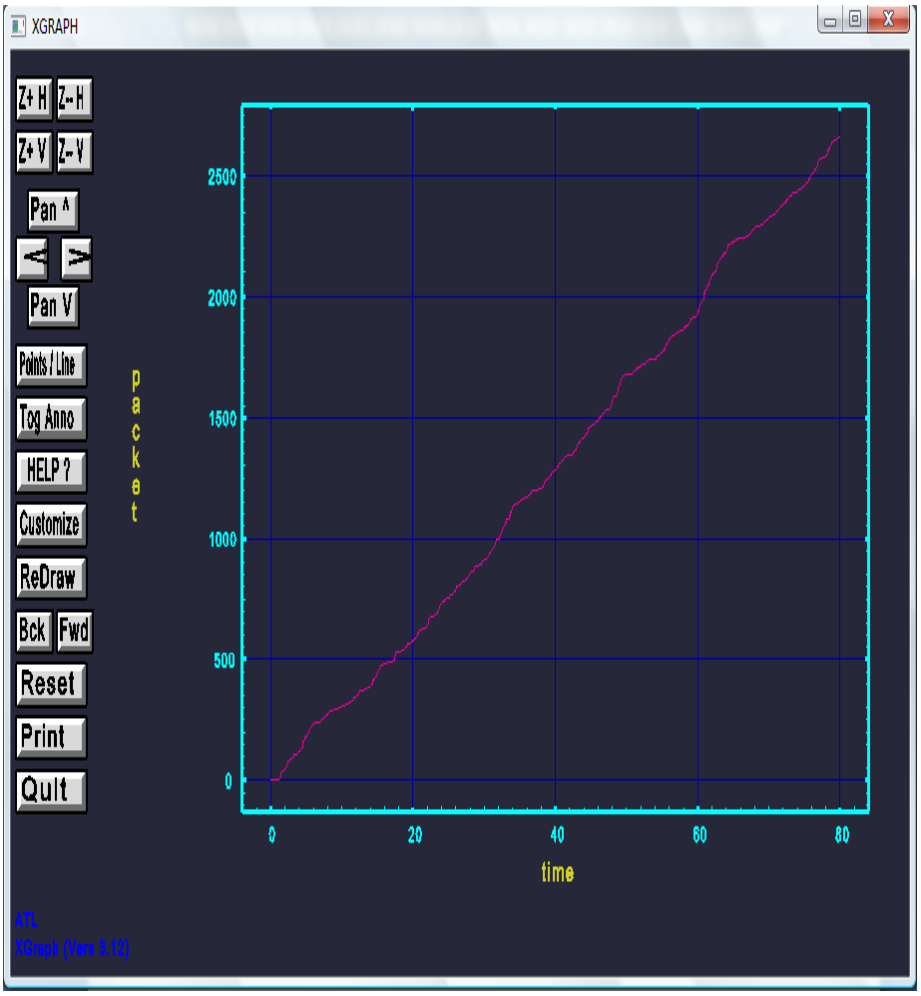

Figure 9: Number of packets transferred

(Figure 10) representing graph with parameter time and packet and showing transfer of packets as per time.

2. Shows Bit Rate

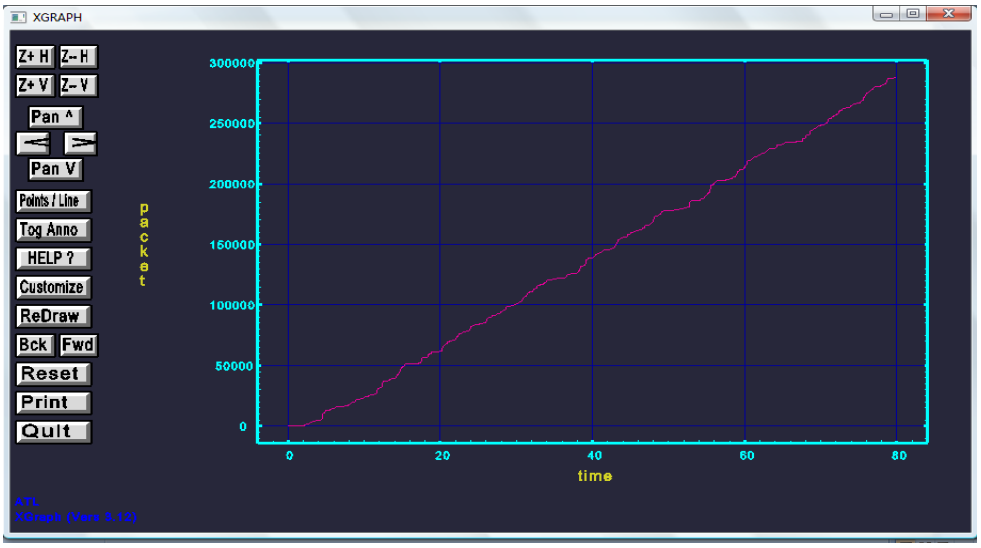

Figure 11: shows Bit Rate

(Figure 11) shows bit rate of packet transferring with the help of two parameters packet and time.

3. Shows Byte Rate

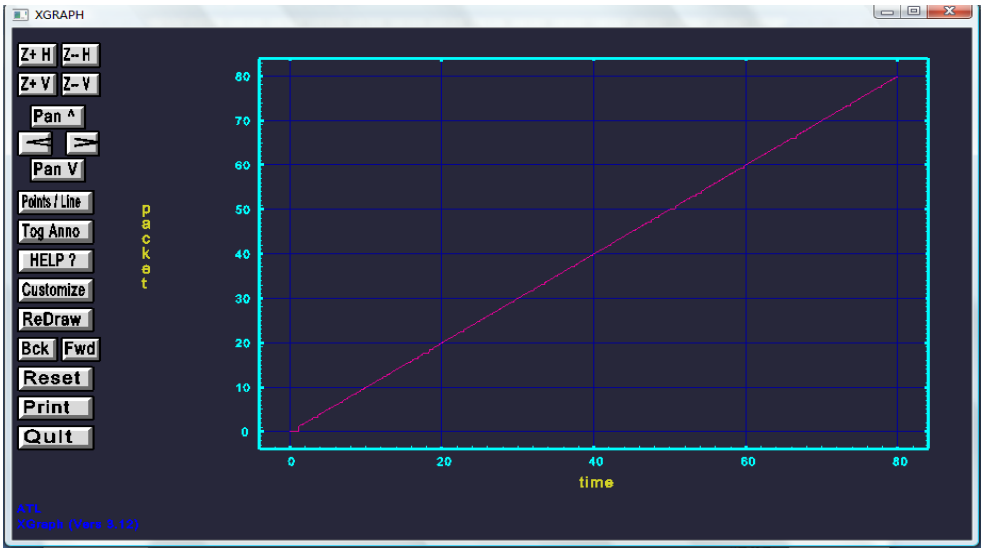

Figure 12: shows Byte Rate

(Figure 12) shows byte rate during packet transferring with the help of two parameters packet and time.

4. Packet Lost

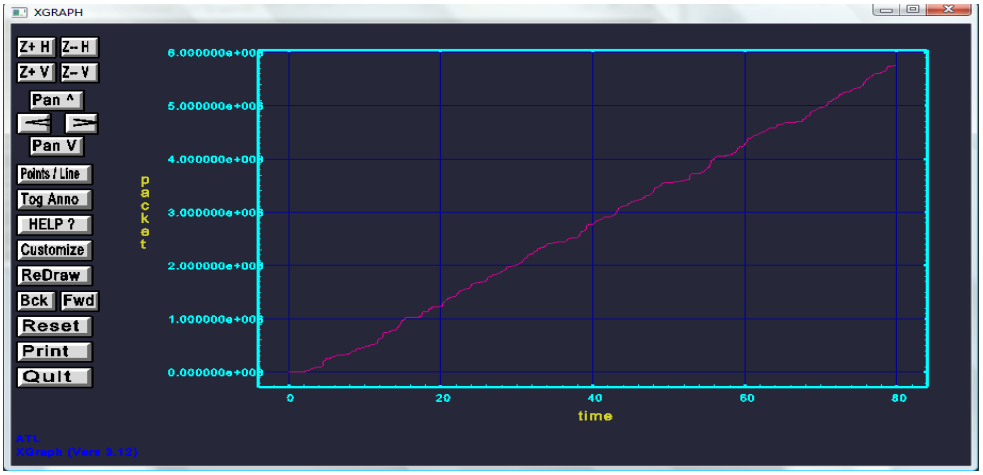

Figure 13: Packet Lost

(Figure 13) shows Packet Lost with the help of two parameters packet and time.

5. Last Packet Time

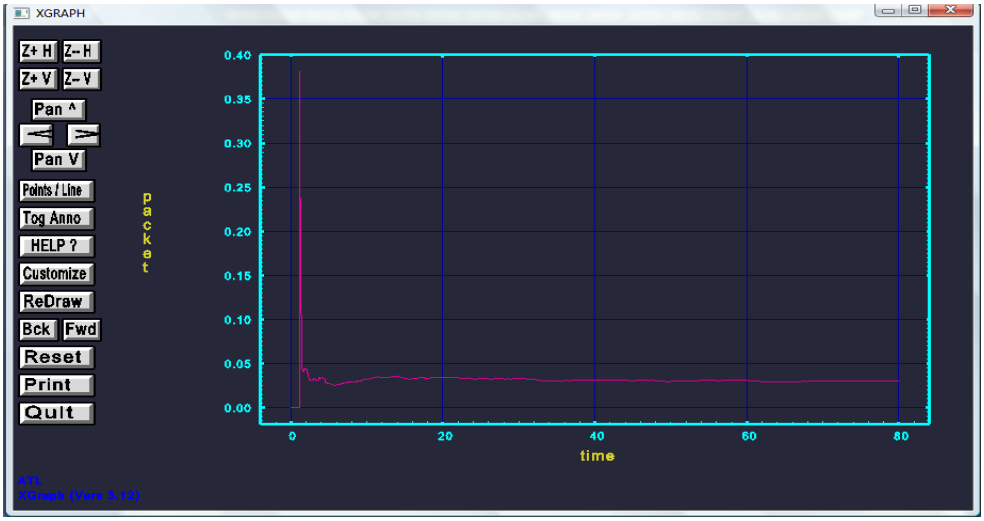

Figure 14: Last Packet Time

(Figure14) shows last packet time with the help of two parameters packet and time.

XIII. CONCLUSION

VANET is an emerging and attractive technology dedicated to safety and comfort services to the vehicle users. Owing to its high dynamic topology and unpredictable channel distribution, it aspires for a suitable routing protocol algorithm that can generate a near seamless network connectivity among the vehicular nodes. In the proposed work we represent an algorithm to solve the congestion problem in all path networks and to get such a path that will provide efficient data transmission over the network. In the network we divide the whole network into sub-networks and we perform the transmission over the sub goal and to achieve the efficient and reliable data transmission.

REFERENCES

[1] J. Blum and L. Hoffman. Challenges of inter vehicle ad hoc networks. IEEE
[2] Transactions on Intelligent Transportation Systems, 5(4):347–351, 2004.

[3] Thomas Clausen , Philippe Jacquet , Anis Laouiti , Pascale Minet , Paul Muhlethaler , Amir Quayyum , and Laurent Viennot , "Optimized link state routing protocol," Internet Draft, draftietf-manet-olsr-05.txt, work in progress, October 2001.
[4] Charles E. Perkins and Pravin Bhagwat, "Highly dynamic destination-sequenced distance vector routing (DSDV)," in Proceedings of ACM SIGCOMM'94 Conference on
[5] Communications Architectures, Protocols and Applications, 1994.
[6] Charles E. Perkins and Elizabeth M. Royer, "Adhoc on-demand distance vector routing," in Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, February 1999, pp. 1405–1413.
[7] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in Mobile Computing, 1996, pp. 153–181.