LOCATION PRIVACY FOR VANET

p- 2172

A PROJECT REPORT

Submitted by

V.JOTHI KIRUTHIKA	71204205014
S.KANMAANI	71204205015
J.SARANYA	71204205042

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2008

P 2172

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"LOCATION PRIVACY FOR VANET"** is the bonafide work of **"V.JOTHI KIRUTHIKA, S.KANMAANI and J.SARANYA"** who carried out the project work under my supervision.

[illegible]

SIGNATURE

Dr.S.Thangasamy, B.E(Hons).,Ph.D.

HEAD OF THE DEPARTMENT

Dept of Information Technology,

Kumaraguru College of Technology,

Coimbatore – 641 006.

[illegible]

SIGNATURE

Mr. K.R.Baskaran B.E., M.S.,

SUPERVISOR

Dept of Information Technology,

Kumaraguru College of Technology,

Coimbatore – 641 006.

The candidates with University Register No 71204205014, 71204205015 and 71204205042 were examined by us in the project viva-voice examination held on....24.04.08......

[illegible]

INTERNAL EXAMINER

[illegible]

EXTERNAL EXAMINER

DECLARATION

We,

V.JOTHI KIRUTHIKA	**71204205014**
S.KANMAANI	**71204205015**
J.SARANYA	**71204205042**

hereby declare that the project entitled " **LOCATION PRIVACY FOR VANET** ", submitted in partial fulfillment to Anna University as the project work of Bachelor of Technology (Information Technology) Degree, is a record of original work done by us under the supervision and guidance of Department of Information Technology, Kumaraguru college of Technology, Coimbatore.

Place: Coimbatore

Date: 24.04.08

[illegible]

[V.Jothi Kiruthika]

Kanmaani.S

[S.Kanmaani]

J. Saranya

[J.Saranya]

Project Guided by

[illegible]

Mr. K.R.Baskaran B.E., M.S.,

ACKNOWLEDGEMENT

We express our sincere thanks to our chairman **Padmabhushan Arutselvar Dr. N. Mahalingam B.Sc, F.I.E** and correspondent **Mr. M. Balasubramaniam, M.Com., MBA** for all their support and ray of strengthening hope extended. We are immensely grateful to our principal **Dr. Joseph V. Thanikal M.E., Ph.D., PDF., CEPIT.,** for his invaluable support to the outcome of this project.

We also thank **Dr.S.Thangasamy B.E(Hons).,Ph.D,** Head of the department of Information Technology for his valuable advice and useful suggestions throughout this project.

We express our heartiest thanks to our project guide and project coordinator **Mr. K.R.Baskaran B.E., M.S.,** Assistant Professor, Department of Information Technology who has helped us to perform our project work extremely well.

We thank the teaching and non-teaching staffs of our Department for providing us the technical support in the duration of our project.

We express our humble gratitude and thanks to our beloved parents and family members who have supported and helped us to complete the project and our friends, for lending us valuable tips, support and co-operation throughout the project work.

ABSTRACT

In vehicular ad hoc networks (VANET), it is possible to locate and track a vehicle based on its transmissions, during communication with the road-side infrastructure or with other vehicles. By tracking a vehicle, it becomes possible to identify the locations visited by the vehicle, thereby, breaching the privacy of the user of the vehicle. Furthermore, the location tracking information about a user can be misused by an adversary. This project aims at providing location privacy for the vehicles in a VANET. To provide location privacy, we employ the grouping of the vehicles in geographical proximity to operate as a single entity. Each group contains a group leader which acts as a proxy for the communication of its members with the road side infrastructure. The group leader periodically collects the group member's information such as the periodic probe data, weather reports, traffic conditions etc and communicates them to the road side infrastructure. Since the vehicle does not directly take part in the communication, it becomes impossible to track the location of a group member, in the case of a global passive adversary. The vehicle's location information remains unrevealed, thereby providing location privacy.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVATIONS

1. RSU	Road Side Unit
2. V2V	Vehicle-to-Vehicle
3. V2I	Vehicle-to-Infrastructure
4. IVC	Inter-Vehicle Communication
5. VANET	Vehicle Ad-hoc NETwork
6. LBS	Location Based Service
7. SUMO	Simulation of Urban Mobility
8. TRANS	TRAffic and Network Simulator

INTRODUCTION

1. INTRODUCTION

1.1 GENERAL:

A mobile ad hoc network (MANET) consists of mobile nodes that connect themselves in a decentralized, self-organizing manner and may also establish multi-hop routes. If the mobile nodes are cars this is called Vehicular Ad Hoc Network (VANET). A **Vehicular Ad-Hoc Network**, or **VANET**, is a form of Mobile ad-hoc network, to provide communications among nearby vehicles and between vehicles and nearby fixed equipment, usually described as roadside equipment. The main goal of VANET is providing safety and comfort for passengers. A special electronic device will be placed inside each vehicle which will provide Ad-Hoc Network connectivity for the passengers. This network tends to operate without any infra-structure or legacy client and server communication. Each vehicle equipped with VANET device will be a node in the Ad-Hoc network and can receive and relay others messages through the wireless network. Collision warning, road sign alarms and in-place traffic view will give the driver essential tools to decide the best path along the way. There are also multimedia, internet connectivity facilities and LOCATION BASED SERVICES (LBS) for passengers, all provided within the wireless coverage of each car. Automatic payment for parking lots and toll collection are other examples of possibilities inside VANET.

1.1.1 Why is Privacy Important for VANETs?

We live in a world where almost any data is available electronically. Most of the future's automobiles will be equipped with navigation systems and therefore technically be able to gather complete movement patterns of its user. All of this would not be much of a problem as long as the car is an isolated system. But future cars will have various communication capabilities. Electronic tolling systems, internet access, maintenance systems, software and media download, off-board navigation systems are just some examples why cars will get connected. A very dangerous and often ignored fact about privacy is that innocent looking data from various sources can be accumulated over a long period and evaluated automatically. Even small correlations of the data may reveal useful information. For instance, the knowledge about specific sensor characteristics may give some hints about the make and the model of car. This in turn may be related to other information to identify a specific car. And once privacy is lost, it is very hard to re-establish that state of personal rights.

1.2 PROBLEM DEFINITION:

Vehicular ad hoc networks (VANET) enable vehicles to communicate among themselves (V2V communications) and with road-side infrastructure (V2I communications). Such networks present various functionalities in terms of vehicular safety, traffic congestion reduction, and location based service (LBS) applications. The unique requirements of maintaining liability of vehicles involved in accidents, and ensuring the safety rendered by the communication between vehicles, challenge the network connectivity, privacy, and certain security in VANET. Advances in localization and tracking techniques enable accurate location estimation and tracking of vehicles in VANET. By tracking a vehicle, it becomes possible to identify the locations visited by the vehicle, thereby, breaching the privacy of the user of the vehicle. Furthermore, the location racking information about a user can be misused by an adversary. Additionally, identifying the LBS applications accessed by vehicle , provides private information of the vehicle's user. Hence the privacy of the user becomes a question mark. Privacy has been recognized as an enabler for VANETs. But we are still lacking appropriate technologies and architectures to accommodate the privacy requirements.

1.3 OBJECTIVE OF THE PROJECT :

In scenarios where a car communicates with a dedicated partner, we assume that in some cases the car's real identity will be required for service usage. In such a case it is obvious that the communication partner has the identity anyway, so the identity must only be protected from the nodes overhearing the communication. We study the problem of privacy breach in a VANET and implement the group navigation to mitigate tracking of a vehicle. We leverage the group to provide communicate among themselves (V2V communications) and anonymous access to LBS applications, and show when such a solution can preserve a vehicle user's privacy.

LITERATURE REVIEW

2. LITERATURE REVIEW

2.1 FEASIBILITY STUDY :

2.1.1 CURRENT STATUS OF THE PROBLEM :

2.1.1.1 VANET SYSTEM MODEL AND ASSUMPTIONS :

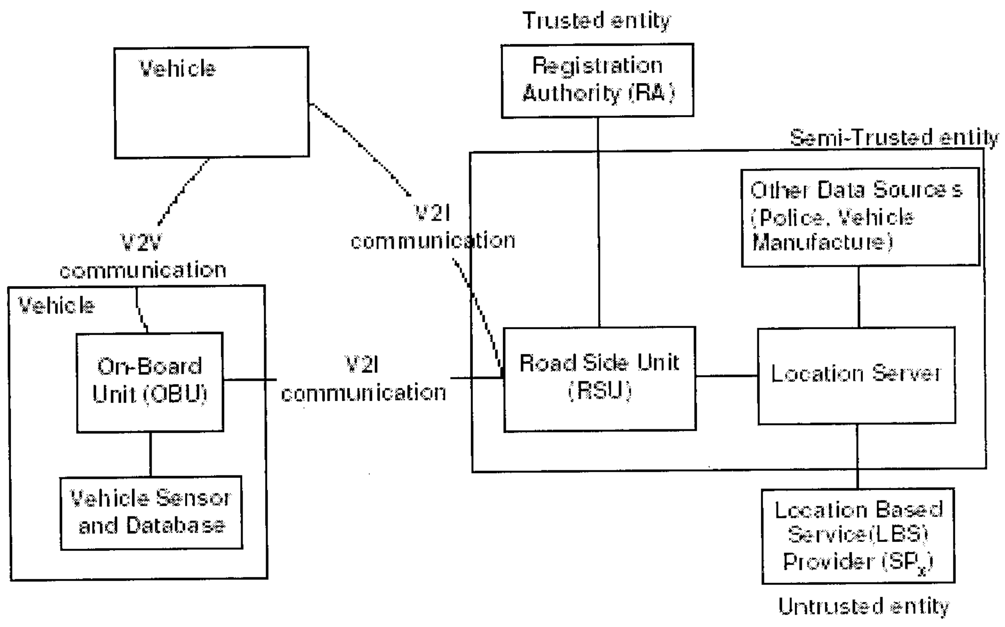


Fig.2.1. Illustration of an inter-vehicle communication system

Fig1 illustrates a typical VANET that consists of vehicles, access points on road side, and a collection of location servers. Vehicles move on roads, sharing collective environmental information between themselves, and with the servers via access points. Fig1 illustrates a detailed view of our system model. A vehicle is enabled with on-board communication unit for V2V and V2I communications, and sensor (for example, GPS) and database units to collect environmental information (for example, location, vehicle speed, tire pressure). The communication unit of the access points is called Road Side Units (RSU), which is connected to location server by a wired network. The location server records all the location data forwarded by the RSUs, and processes the data together with information from other data sources for example, vehicle manufacturers, police, traffic management center, weather information center. The location server also provides an interface for the location based Service Providers (SP). In addition, a trusted registration Authority (RA) provides authentication and authorization service to both vehicles and LBS providers. As in we assume that a suitable public key infrastructure is available in the VANET. Before joining the VANET, each vehicle registers with the trusted RA. We also assume that each LBS service provider registers with the RA, and obtains a public/private key pair. During registration, each vehicle i is pre-loaded with a public / private key pair ($K_{PID\ i}$ $K^{1}_{PID\ i}$), and a corresponding public key certificate signRA($K_{PID\ i}$). Each vehicle also registers for any location based service application that is of interest.

2.1.1.2 Trust Assumptions and Adversary Model

We assume that the registration authority (RA) is a trusted entity in our model, as shown in fig1. The infrastructure including the RSUs and the location server are only semitrustede to operate as expected. We additionally, assume that the RSUs are able to estimate location of a vehicle based on the vehicle's transmission signal. In our model, we assume a global passive adversary. Such an adversary is able to overhear all the broadcasts of all the vehicles, and hence, able to estimate their locations. These applications obtain and make use of the most recent location of a mobile node, in order to provide a requested service. For example, the service may be a query by a vehicle to find the nearest shopping mall to its current location.

2.1.1.3 Application Scenarios Considered

We consider three typical classes of VANET applications, cooperative driving, probe vehicle data, and location based service (LBS). In the cooperative driving application, adequate equipped vehicles maintain a very short separation (intra-convoy spacing) between each other and move smoothly with the same pre-defined speed (convoy speed). These vehicles communicate with each other frequently either directly or via communication equipments on road side. For example, vehicles broadcast their status information (e.g. speed, location, acceleration) every 500 ms. the advantage of cooperative driving is the increase in both safety and highway capacity resulting from the automation and close coordination of vehicles.

2.1.1.4. Relevant Constraints of VANET

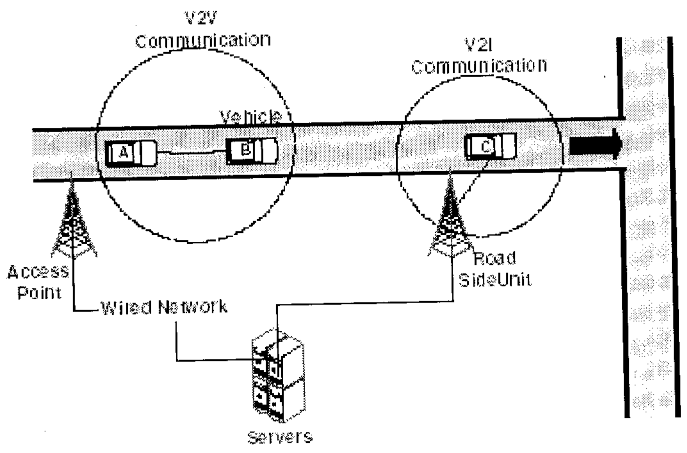


Fig. 2.2. Illustration of V2V and V2I communication

VANET poses constraints such as in mobility of vehicles, and in safety application requirements. The mobility of vehicles can be observed to have the following unique characteristics: (1) The movement of vehicles is spatially restricted. For example, as illustrated in Fig. 2, the movement of vehicles is restricted to be in lanes, in both streets and freeways. (e) The vehicles are spatially dependent on each other in movement. For example, as illustrated in Fig. 2, a succeeding vehicle A (following) must keep a minimum safety distance from a preceding vehicle B (being followed). The safety applications impose constraints in terms of the maximum between two safety message broadcasts in cooperative driving, and maximum period between two replies in probe data. Therefore, overall, any location privacy enhancement scheme designed for VANET must take into account these unique constraints.

2.1.1.5 Privacy Threats

There are a couple of situations, where privacy should be discussed. It has to be decided which degree of privacy is necessary under given circumstances and the system has to be designed accordingly. Here are some examples for the problems we have to tackle in a widespread VANET.

– The police use hello beacons to calculate driving behavior and issues speeding tickets

– An employer is overhearing the communications from cars on the company parking lot. After distinguishing which car-identifier belongs to which employee he automatically gets exact arrival and departure dates.

– A private investigator easily follows a car without being noticed by extracting position information from messages and hello beacons.

–Insurance companies gather detailed statistics about movement patterns of cars. In some cases individual persons may be charged for traffic accidents based on gathered movement patterns.

–A criminal organization has access to stationary communication boxes and uses the accumulated information to track law enforcement vehicles. The same technique could be used by a foreign secret service to track VIPs.

As we can see from these examples, most issues are related to position and identifiers. More specifically, either keeping identifiers and relating them to other received identifiers (re-recognition) or correlating the identifier with a real-world identity (identification). Analyzing the relations of various position-identifier pairs, a multitude of attacks on privacy can be carried out, where the given examples represent only a small subset. The first example might be the easiest to resolve, because in most countries a defendant is innocent until it can be proven that he is guilty. In our case that means that the police must be able to prove that the Hello beacons are used in mobile ad-hoc networks to maintain information about the nodes' neighborhood. Culprit is really the one who was speeding. So instead of using one's original identity in the system, a pseudonym may be used. Unless there is no perfectly provable mapping between the pseudonym and real-world identity, the police would have a hard time issuing a ticket. In the second example this may not be enough, because the employer has other means of correlating real-world identities and car-identifiers. And he may guess as well. In this case, it would be desirable to change the car's identifiers from time to time. In the third example however, even these precautions would not be sufficient. To prevent being followed, the car's identifier would have to be changed while moving. One possible approach is geobound pseudonyms. Note that a concept considering changing identifiers on application layer also means that all lower layers must change their addresses / identifiers at least as often as application layer IDs. This would require frequent changes of MAC-addresses and network addresses for instance. Some simple experiments with our prototypes have shown that this is doable in principle, but remains an extremely challenging task for large systems. In addition this will definitely decrease overall performance due to collisions and/or increased signaling overhead. In scenarios where a car communicates with a dedicated partner, we assume that in some cases the car's real identity will be required for service usage. In such a case it is obvious that the communication partner has the identity anyway, so the identity must only be protected from neighbors overhearing the communication.

2.1.1.6 Research Projects:

Currently, there are some car-to-car network research projects which have operational prototypes, such as FleetNet, Carisma and VSC. However, only the VSC project dealt with security, where privacy has only been a minor issue. Nevertheless, security and privacy are enabling technologies for the applications those projects envision. The Car-to-Car Communications Consortium, which has been founded recently, aims at standardizing car-to-car communications. It strongly encourages privacy enhancing concepts in this context. Other activities, such as IEEE's P1556 working group, have been raising security and privacy concerns and standards bodies like ISO's TCe04/WG16 are discussing privacy now.

2.1.2 PROPOSED SYSTEM AND ADVANTAGES:

2.1.2.1 Use of Group Concept to Avoid Overhearing:

We make the following observations that motivate the group concept applied in our solution.

1) Vehicles in geographical proximity often share redundant information such as road and traffic conditions. Hence, in V2I based applications, such as probe vehicle data, where the vehicles respond to requests received from the infrastructure, not all vehicles need to send replies.

P. 2179

2) The mobility of vehicles is spatially restricted and spatially dependent. Hence, vehicles in geographical proximity can navigate as a group, with the same average velocity due to the spatial dependency, and with similar direction due to the spatial restrictions, over a period of time.

We make use of the above observations, and propose to enable vehicles to form a group. In order to form a group, we restrict the vehicles to be in a group if each group member can hear broadcasts of every other group member. Since vehicles in a group will move relative to each other, and on average have the same velocity, a group can be represented by a single vehicle that we refer to as the group leader. Then for most of the V2I communication based VANET applications, it is sufficient if only the group leader communicates on behalf of the group. Consequently, the remaining vehicles in the group are able to remain silent for an extended period of time that is bounded by the time they remain in the group and hence their identity is preserved.

An extended silent period can enhance the location privacy provided to a vehicle. Therefore, for VANET applications not requiring all vehicles to broadcast, i.e. for applications not requiring very frequent safety message broadcasts from the vehicles, we can increase level of anonymity by employing groups. We consider the probe vehicle data application, where typically, the vehicles send probe replies once in several tens of seconds. By using vehicular groups, we offer the following benefits:

(1) The silent period of a group member vehicle is extended, if the vehicle does not change group between two probe data requests.

(2) Unnecessary overhead and redundancy of the neighboring vehicles broadcasting possibly redundant probe data is reduced, since only the group leader replies to the RSU with probe data.

However, for safety applications such as cooperative driving, where all vehicles broadcast at a high frequency, the group benefits are not fully realizable. This is because, each vehicle must broadcast its location, speed, and other spatial parameters for safety, as well as to maintain liability. Hence, under the performance bottleneck of the small safety broadcast period, the advantageous applicability of vehicular group in mitigation of tracking is limited. Nevertheless, vehicular groups can be leveraged to defend against threats on privacy when accessing LBS applications.

2.2 HARDWARE REQUIREMENTS (MINIMUM) :

Processor	:	Pentium IV
Speed	:	Above 500 MHz
RAM capacity	:	128 MB
Floppy disk drive	:	1.44 MB
Hard disk drive	:	20 GB
Key Board	:	108 keys
Mouse	:	Optical Mouse
CD Writer	:	Optional
Printer	:	Optional
Motherboard	:	Intel
Monitor	:	17”

2.3 SOFTWARE REQUIREMENTS:

- SUMO Simulator – to simulate urban vehicular traffic.
- NS2 Simulator – to simulate the vehicular adhoc network.
- TRANS – Traffic and Network Simulator to integrate SUMO and NS2.

- Java Development Kit 6.0.
- CYGWIN/X – LINUX like environment for WINDOWS.

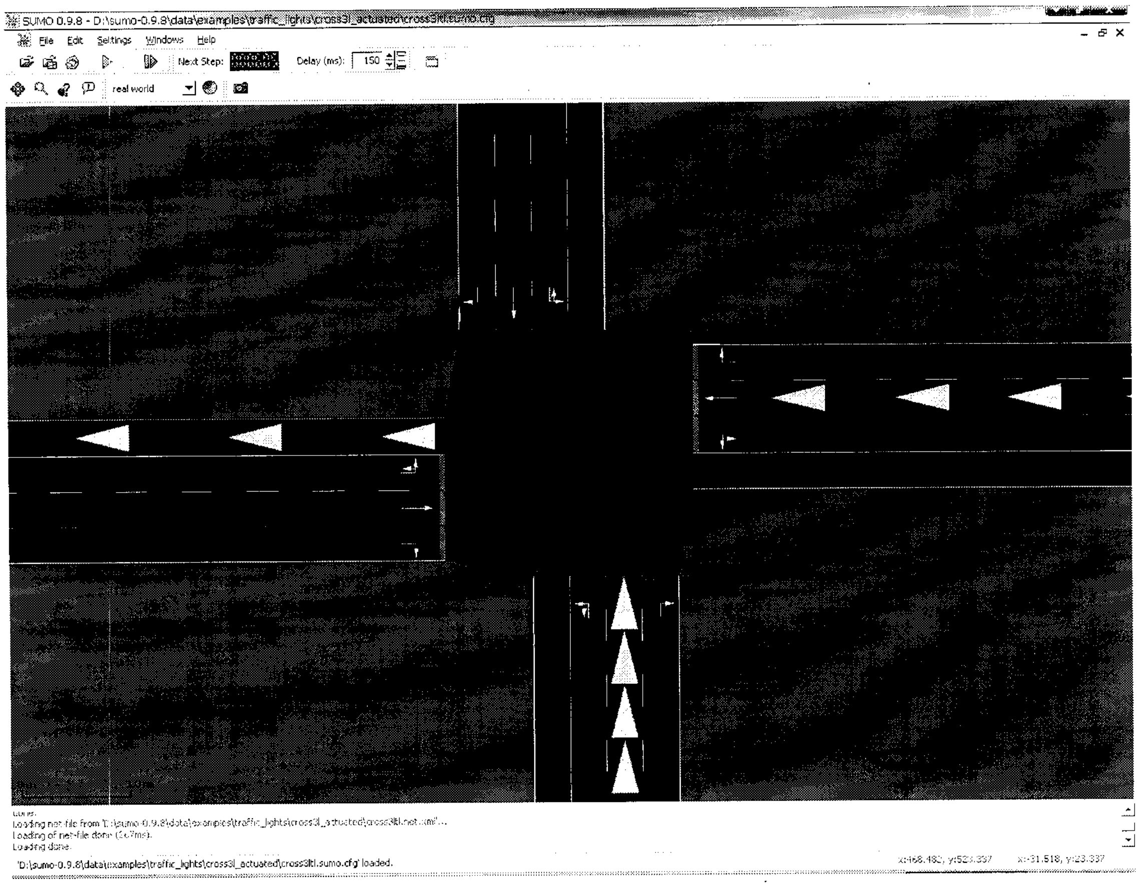
2.4 SOFTWARE OVERVIEW:

2.4.1 SUMO SIMULATOR :

Sumo stands for **Simulation of Urban MObility.** SUMO is a traffic simulation tool. It was primarily designed for urban street networks, but it may also be used for highway traffic simulations. In the near future it will be extended to model other transit modes simultaneously with ordinary car traffic.

2.4.1.1 Features :

- High portability (using standard - c++ and portable libraries only)
- Collision free vehicle movement
- Different vehicle types
- Single-vehicle routing
- Multi-lane streets with lane changing
- Junction-based right-of-way rules
- Hierarchy of junction types
- A fast openGL graphical user interface
- Dynamic routing


SUMO 0.9.8 - D:\sumo-0.9.8\data\examples\traffic_lights\cross3l_actuated\cross3ltl.sumo.cfg
File
Edit
Settings
Windows
Help
Next Step:
Delay (ms): 150
real world

```
done.
Loading net-file from 'D:\sumo-0.9.8\data\examples\traffic_lights\cross3l_actuated\cross3ltl.net.xml'...
Loading of net-file done (3.7ms).
Loading done.
'D:\sumo-0.9.8\data\examples\traffic_lights\cross3l_actuated\cross3ltl.sumo.cfg' loaded.
x:468.482, y:523.337     x:-31.518, y:23.337
```

Fig 2.3 CARAVAN SIMULATION

In sumo a street network consists of nodes (junctions) and edges (streets connecting the junctions). Thus, if we want to create a network with two streets, subsequent to each other, we need three nodes and two edges.

2.4.1.2 Creating Nodes :

All nodes have at location (x- and y-coordinate, describing distance to the origin in meters) and an id for future reference. Thus our simple node file looks as follows

```
<nodes>
<node id="1" x="-500.0" y="0.0" />
<node id="2" x="+500.0" y="0.0" />
```

```
<node id="3" x="+501.0" y="0.0" />
</nodes>
```

You can edit a file with a text editor of your choice and save this for instance as hello.nod.xml where .nod.xml is the default suffix for Sumo node files.

2.4.1.3 Creating Edges :

Now we are connecting the nodes with edges. This is as easy as it sounds, we have a source node id, a target node id, and an edge id for future reference. Edges are directed, thus every vehicle traveling this edge will start at fromnode and end at tonode.

```
<edges>
<edge fromnode="1" id="1to2" tonode="2" />
<edge fromnode="2" id="out" tonode="3" />
</edges>
```

Save this data into a file called hello.edg.xml. Now that we have nodes and edges we can call the first sumo tool to create a network.

2.4.1.4 Generating Network :

After making sure netconvert is somewhere in our PATH and calling

```
netconvert --xml-node-files=hello.nod.xml --xml-edge-files=hello.edg.xml --
output-file=hello.net.xml
```

This will generate our network called hello.net.xml.

2.4.1.5 Creating Routes :

Now we have a net, we still need a car. In SUMO the vehicles have types defining their basic properties such as length, acceleration and deceleration and maximum speed. Furthermore it needs a so called sigma parameter which introduces some random behavior and is due to the car following model used. Setting it to 0 gives a deterministic car.

Now we define a route for our car which simply consists of the two edges we defined. The reason why we need two edges is that in SUMO the car disappears as soon as it has reached the last edge of its route. Last but not least we define our single car mainly referring to the entries before and giving it a departure time as in the following hello.rou.xml file.

```
<routes>
<vtype
accel="1.0"decel="5.0"id="Carlength="2.0"maxspeed="100.0"sigma="0.0" />
<route id="route0">1to2 out</route>
<vehicle depart="1" id="veh0" route="route0" type="Car" />
</routes>
```

2.4.1.6 Creating Configuration :

Now we glue everything together into a configuration file

```
<configuration>
<files>
<net-file>hello.net.xml</net-file>
<route-files>hello.rou.xml</route-files>
```

```
</files>
<simulation>
<begin>0</begin>
<end>10000</end>
</simulation>
</configuration>
```

Save this to hello.sumo.cfg and can start the simulation by either

```
sumo -c hello.sumo.cfg  or
guisim -c hello.sumo.cfg
```

2.4.2 NS2 SIMULATOR:

2.4.2.1 About NS-2

NS-2 is an open-source simulation tool running on Unix-like operating systems. It is a discreet event simulator targeted at networking research and provides substantial support for simulation of routing, multicast protocols and IP protocols, such as UDP, TCP, RTP and SRM over wired, wireless and satellite networks. It has many advantages that make it a useful tool, such as support for multiple protocols and the capability of graphically detailing network traffic. Additionally, NS-2 supports several algorithms in routing and queuing. LAN routing and broadcasts are part of routing algorithms. Queuing algorithm includes fair queuing, deficit round robin and FIFO. NS-2 started as a variant of the REAL network simulator in 1989. REAL is a network simulator originally intended for studying the dynamic behavior of flow and congestion control schemes in packet-switched data networks. In 1995 ns development was supported by Defense Advanced Research Projects Agency DARPA through the VINT project at LBL, Xerox PARC, UCB, and USC/ISI. The wireless codes from the UCB Daedelus and CMU Monarch projects and Sun Microsystems have added the wireless capabilities to ns-2. Valery Naumov proposed a list-based improvement for ns-2 involving maintaining a double linked list to organize mobile nodes based on their X-coordinates. When sending a packet, only those neighbor nodes are considered, which are within a circle corresponding to the carrier-sense threshold energy level, below which a node cannot hear the packet. Compared to the original version, where all nodes in the topology are considered, its considerable gain in run-time, performance goes down by about 4 to 20 times, depending on the size of the topology. The larger the topology and greater the number of nodes, the greater is the improvement seen with the list-based implementation. NS-2 is available on several platforms such as FreeBSD, Linux, SunOS and Solaris. NS-2 also builds and runs under Windows with Cygwin. Simple scenarios should run on any reasonable machine; however, very large scenarios benefit from large amounts of memory and fast CPU's.

2.4.2.2 NS-2, implementing languages

NS-2 is basically written in C++, with an OTcl (Object Tool Command Language) interpreter as a front-end. It supports a class hierarchy in C++, called compiled hierarchy and a similar one within the OTcl interpreter, called interpreter hierarchy. Some objects are completely implemented in C++, some others in OTcl and some are implemented in both. For them, there is a one-to-one correspondence between classes of the two hierarchies. But why should one use two languages? The simulator can be viewed as doing 2 different things. While on one hand detailed simulations of protocols are required, we also need to be able to vary the parameters or configurations and quickly explore the changing scenarios. For the first case we need a system programming language like C++ that effectively handles bytes, packet headers and implements algorithms efficiently. But for the second case iteration time is more important than the run-time of the part of task. A scripting language like Tcl accomplishes this.

2.4.2.3 Architecture of ns-2

As already mentioned above, ns-2 is an object-oriented, discrete event simulator. There are presently five schedulers available in the simulator, each of which is implemented by using a different data structure: a simple linked-list, heap, calendar queue (default) and a special type called "realtime". The scheduler runs by selecting the next earliest event, executing it to completion, and returning to execute the next event. The units of time used by the scheduler are seconds. An event is handled by calling the appropriate Handler class. The most important Handler is NsObject with TclObject as its twin in the OTcl world. They provide all the basic functions allowing objects to interact one with another. For this purpose the receive function group is mainly used. For handling OTcl statements in C++ NsObjects provide the so-called command function. NsObject is the parent class for some important classes as the Classifier, the Connector and the TraceFile class.

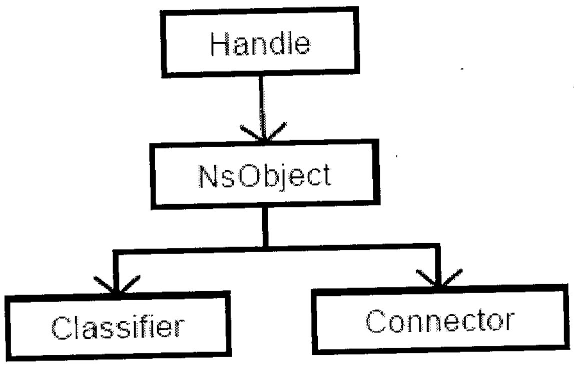


Fig 2.4 Class Diagram of NS2

2.4.2.4 Usage of ns-2

An ns-2 simulation is controlled by a TCL scripts, which contains all necessary parameters and configurations. Additionally the ′opt′ parameters within the TCL script can be modified from the command line as shown below.

```
ns script.tcl -nn 100 -x 5000 -y 5000 -stop 800 \
-tr out.tr -sc mov -cp traffic
```

The TCL script specifies the path of movement and connection files to be loaded as well as the path to the trace files, usually a nam and a tr file, which are the product of a simulation.

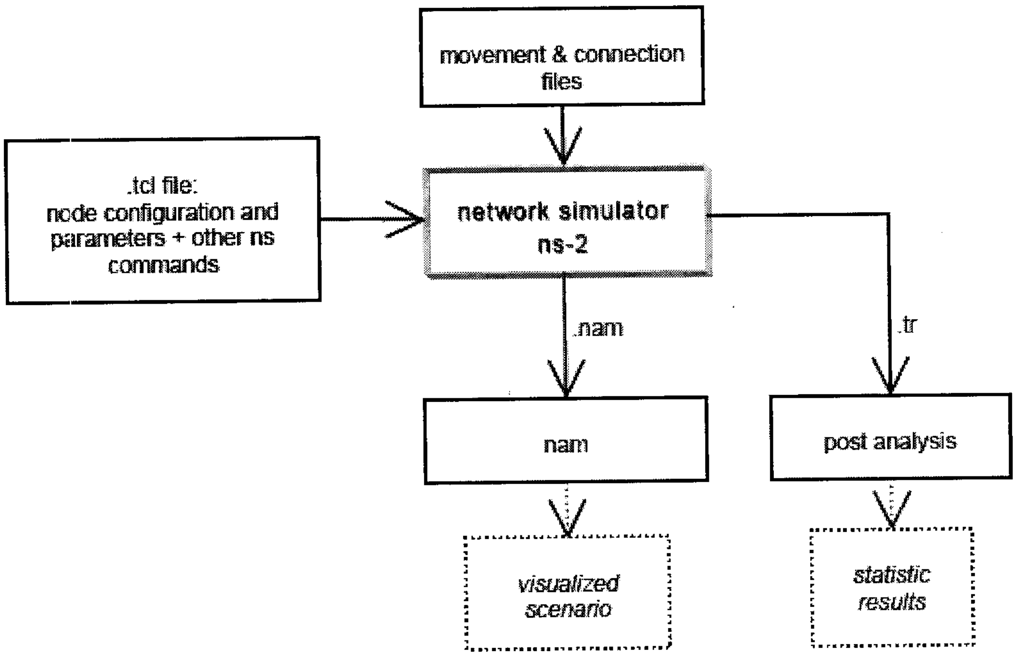

movement & connection files
.tcl file: node configuration and parameters + other ns commands
network simulator ns-2
.nam
.tr
nam
post analysis
visualized scenario
statistic results

Fig 2.5. NS2 Usage Representation

2.4.2.5 TCL simulation scripts

As already mentioned a TCL script is used for configuring and parameterzing a simulation. It consists of several important parts:

□Physical and protocol specifications

□Node creation and movement (mostly imported from a separate file, the so called movement, scene or scenario file)

□Node communication (mostly imported from a separate file, the so called traffic, connection or communication file)

2.4.2.6 Trace files

One gets two trace files as a result of a simulation: a normal trace file (created by $ns_ trace-all commands) and a nam trace file ($ns_ nam-trace all). The nam trace file is a subset of a normal trace file with the suffix ".nam". It contains information for visualizing packet flow and node movement for use with the homonymous nam tool. Nam is a Tcl/TK based animation tool for viewing network simulation traces and real world packet trace data. The design theory behind nam was to create an animator that is able to read large animation data sets and be extensible enough, so that it could be used in different network visualization situations. For studying protocol behavior one has to refer to the normal trace file with the suffix ".tr". It contains all requested trace data produced by a simulation. In the TCL configuration script one can tell the simulator which kind of traces information shall be printed out: agent, route and mac trace. They can separately be turned on or off for every node

```
$ns_ node-config -agentTrace ON/OFF
$ns_ node-config -routerTrace ON/OFF
$ns_ node-config -macTrace ON/OFF
```

2.4.2.7 Limitations of ns-2

A simulator model of a real-world system is necessarily a simplification of the real-world system itself. Especially for WLAN simulations one has to be very generous, because there are tremendously many quick changing parameters, which cannot be handled up to now and in the near future. There are also some practical limitations. We found out that the maximum number of mobile nodes, ns-2 allows, is 16250 (nn_max). Processor speed is also a problem. For example, on a 2 GHz machine a 500 node simulation, with the random way point movement model, in an area of 5x1 km2 with 100 connections over 600 seconds, easily takes several days and requires easily a GB of RAM.

2.4.3 TRANS (Joint TRAffic and Network Simulator for VANET):

2.4.3.1 TraNS Configuration:

The application is divided in two modules. The first module manages the traffic simulator (SUMO) and the second module manages the network simulator (NS2). The former, can run either on a Windows or Linux machine where SUMO is installed. The later should be executed in the machine where NS2 is installed (typically Linux).

You can run the TraNS application in two ways - either from the command line or directly with the jar file. If you decide to use the command line, go to the folder where the TraNS_0.21.jar file is and type the following:

java -jar TRANS_0.21.jar

If we plan to generate mobility traces using very large road network topology files like those imported from TIGER database, TraNS has to be run from the command line, since it is necessary to increase the java heap size. The command to issue is:

java -Xms512m -Xmx1024m -jar TRANS_0.21.jar

In this case the parameter -Xms is the memory size from which the application starts and -Xmx is the memory limit usage. When the TraNS application is started (using either command line or graphical mode) then the selection window will appear. Select which module needed : NS2, SUMO or both.

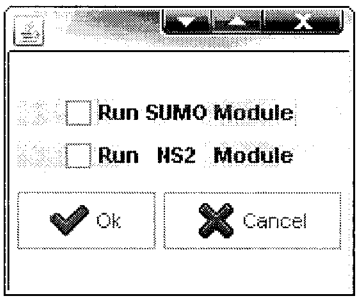
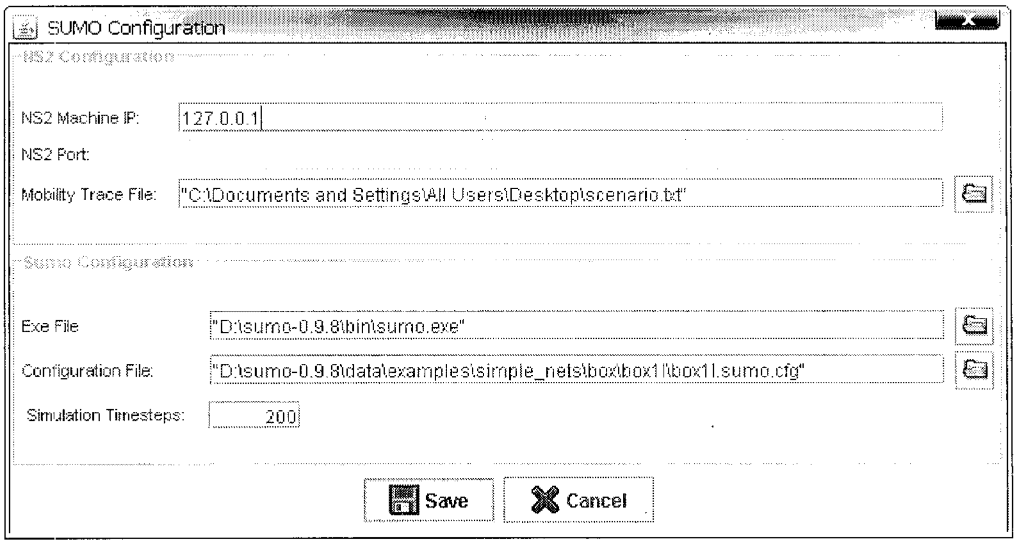


Fig 2.6 Sumo Startup

In case if its needed only to generate the mobility traces, just run the SUMO side module. However, we also have the option to send these traces once they are generated to NS2 simulator and run the corresponding VANET simulation automatically. Once the application is running, we have to configure the communication between the two modules, the filenames used during the simulation and the parameters of simulation.

2.4.3.1.1 SUMO module


SUMO Configuration

NS2 Configuration

NS2 Machine IP: 127.0.0.1

NS2 Port:

Mobility Trace File: "C:\Documents and Settings\All Users\Desktop\scenario.txt"

Sumo Configuration

Exe File "D:\sumo-0.9.8\bin\sumo.exe"

Configuration File: "D:\sumo-0.9.8\data\examples\simple_nets\box\box1l\box1l.sumo.cfg"

Simulation Timesteps: 200

Fig 2.7 Configuring Sumo module

First, within the SUMO module, open the *File/Configuration* menu. In this module, one should specify the *IP address* of the host on which NS2 is running. By default the NS2 module uses port 4444 for communication with the SUMO module. However, it is possible to change this port directly in the configuration file (./files/trans.cfg). *The Mobility Trace File* field specifies the path to the file which will contain mobility traces generated by SUMO and parsed by TraNS. This file will be ready to use as a simulation scenario in NS2 simulation.

Afterwards, one has to specify the directory in which the SUMO executable file is (the *Exe File* field). In order to generate the mobility traces one has to first generate the network and routes files. The former define the road network topology, whereas the latter specify routes of the simulated vehicles. Information about the network and route files is stored in a SUMO configuration file (*.sumo.cfg) and the path to this file has to be specified in the *Configuration File* filed. In the *Simulation Timesteps* field one has to specify the duration of the simulation.

2.4.3.1.2 NS2 module

Configure the NS2 module - *File/Configuration Options* menu . First, it is possible to run NS2 right after the mobility traces are generated by simply checking the corresponding check box. If you enable this option, you have to specify the command to run NS2 in the *Exe File* field. Next, select the simulation file we want to run with NS(TCL File field). Be sure that this file contains a reference to the mobility trace file location. The same reference should be specified in the *Mobility Trace File* field.

2.4.3.2 TRANS ARCHITECTURE :

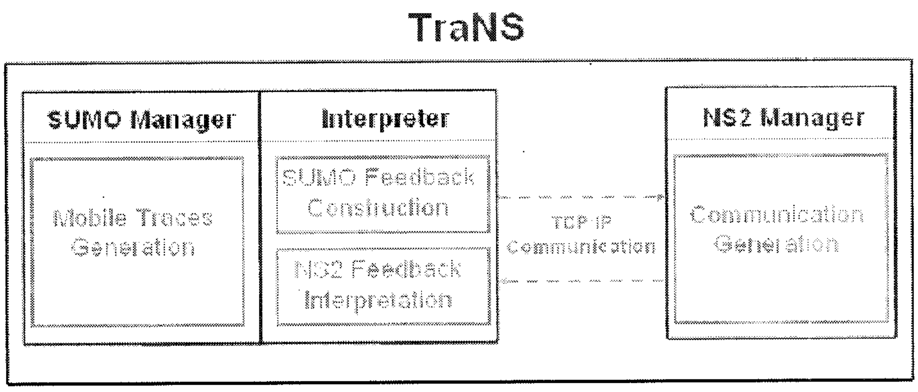


Fig 2.8 TRANS Architecture

Fig 2.9 Conversin of Vehicles in SUMO Simulation to NS2 nodes

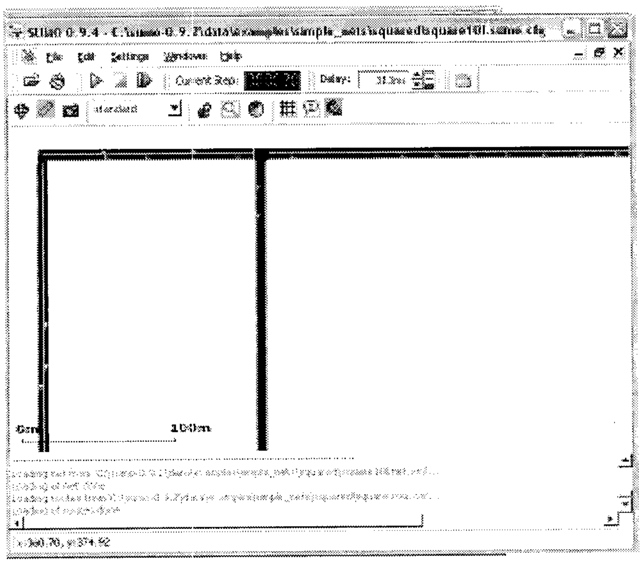


Fig 2.9 a) VEHICLES IN SUMO SIMULATION

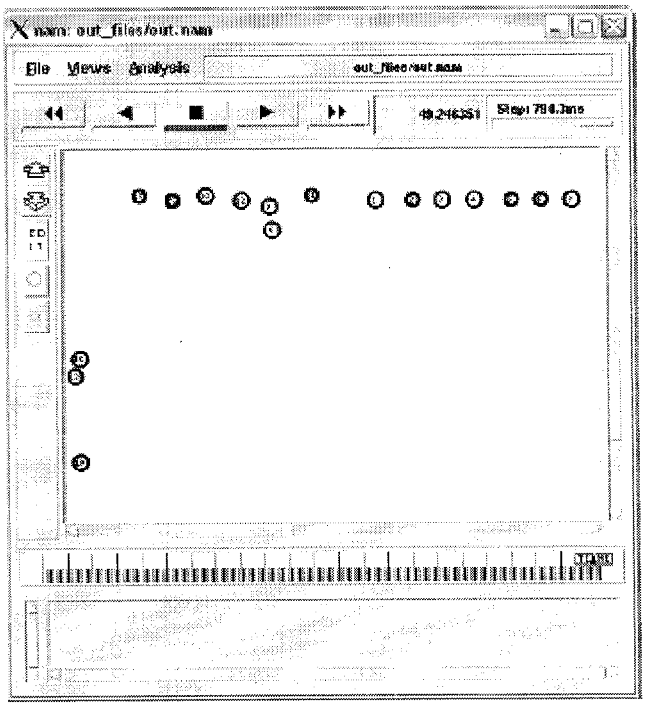


Fig 2.9 b) NODES IN NAM

2.4.3.3 Generation of Mobility Traces

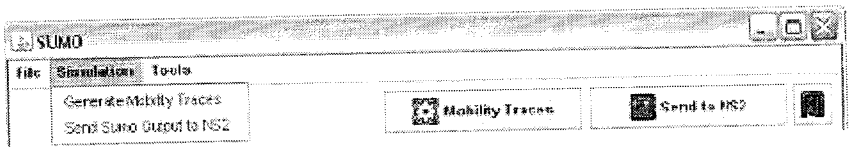


Fig 2.10 Generation of Mobility Traces

When you succeed with the configuration, you are ready to generate mobility scenario for NS2. Click on the *Mobility Traces* button or select Simulation/Generate Mobility Traces menu, which will run the SUMO simulation and will generate the scenario file for NS2 simulation.If you are running NS2 module on a different machine, you can send this scenario file by clicking on the *Send to NS2* button.

2.4.4 CYGWIN/X :

2.4.4.1 CYGWIN :

Cygwin is a collection of tools originally developed by Cygnus Solutions to allow versions of Microsoft Windows to behave in ways familiar to Unix users. Programs supported by Cygwin work well on Windows NT, Windows 2000, Windows XP, Windows Vista and some run acceptably on Windows 9x.Cygwin consists of

o A library that implements the POSIX system call API in terms of Win32 system calls

o A GNU development toolchain (such as GCC and GDB) to allow basic software development tasks o A large number of application programs equivalent to those on the Unix system.

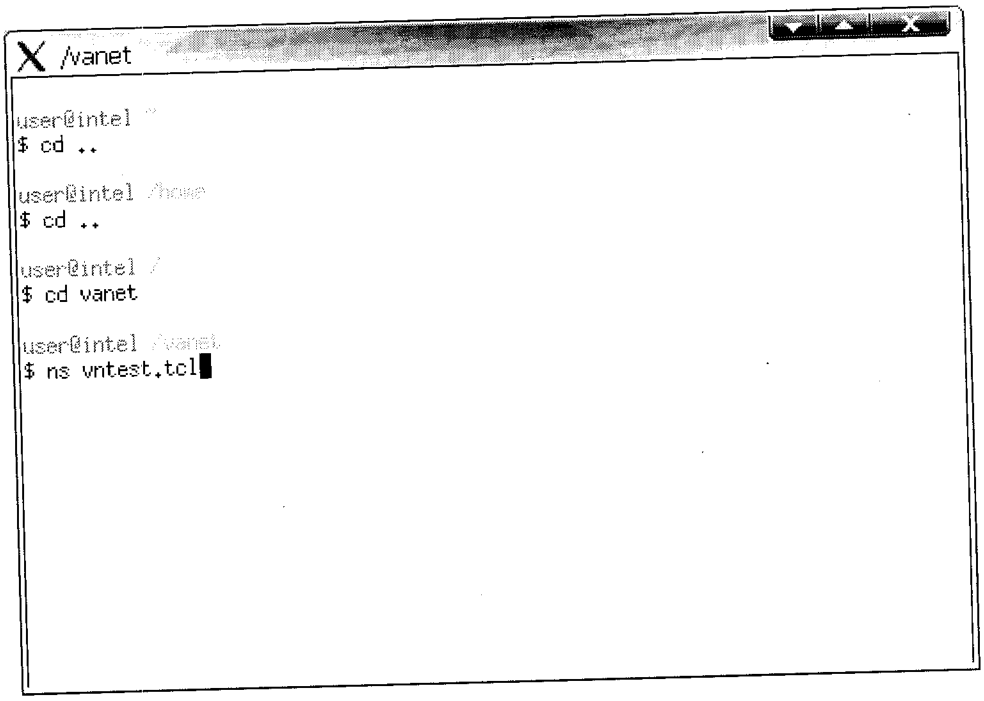
Many Unix programs have been ported to Cygwin, including the X Window System, KDE, GNOME, Apache, and TeX. Cygwin permits installing inetd, syslogd, sshd, Apache, and other daemons as standard Windows services, allowing Microsoft Windows systems to work like Unix and Linux servers. All of these programs are installed by running Cygwin's "setup" program, which downloads the necessary files from the Internet. Setup can install, update, and remove programs and download the source code for them.

While Cygwin provides header files and libraries that make it possible to recompile or to port Unix applications for use on computers running Microsoft Windows operating systems, it does not make binaries compatible with such computers running without Cygwin.

2.4.4.2 CYGWIN/X :

Cygwin/X is a port of the X Window System to the Microsoft Windows family of operating systems. Cygwin/X runs on all recent consumer and business versions of Windows; as of 2003-12-27 those versions are specifically Windows 95, Windows 98, Windows Me, Windows NT 4.0, Windows 2000, Windows XP, and Windows Server 2003.

Cygwin/X consists of an X Server, X libraries, and nearly the entire standard X clients, such as xterm, xhost, xdpyinfo, xclock, and xeyes. Cygwin/X, as the name implies, uses the Cygwin project which provides a UNIX-like API to Xlib and X clients, thereby minimizing the amount of porting required.



```
user@intel ~
$ cd ..

user@intel /home
$ cd ..

user@intel /
$ cd vanet

user@intel /vanet
$ ns vntest.tcl
```

Fig 2.11 Cygin Simulation Environment

METHODOLOGY USED

3 METHODOLOGY USED:

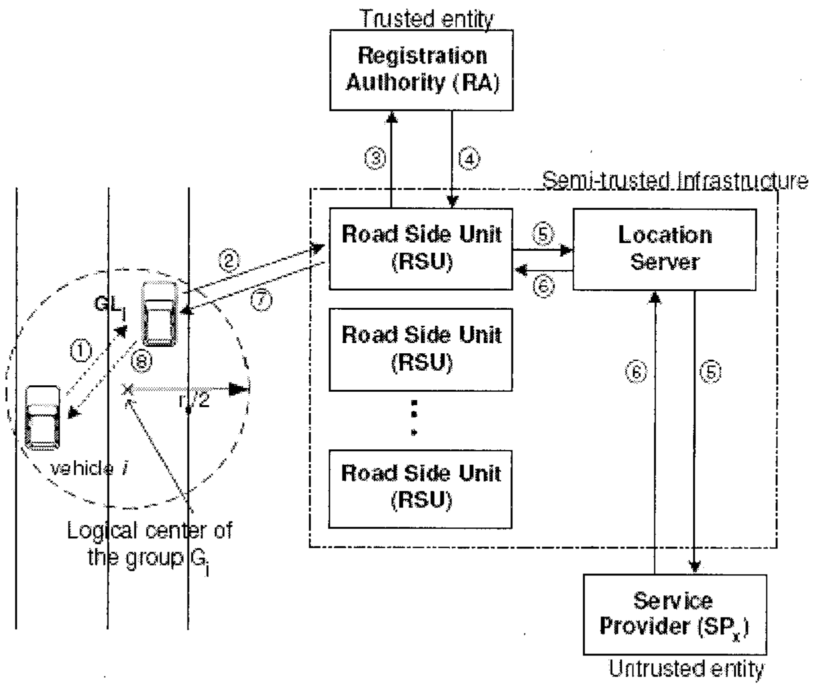


Fig 3.1 Illustration of the anonymous access to LBS application.

In our implementation, the vehicle accessing the LBS application can make use of the group leader as a proxy for anonymous access. We describe this case below.

3.1 Protocol description:

In Fig 3.1 upon receiving the application request from vehicle i (in Step 1), the group leader GL_j of i's group G_j forwards the request with its own address, to the registration authority RA via the RSU (in Step 2-3). The RA validates the application request, and then provides a session key $k_{x,i}$ to both the service provider (SP_x) and vehicle i (Step 4-7). This key is used to encrypt the entire communication that takes place between i and the SP_x. GLj broadcasts the communication received from SP_x (via RSU) to the group (Step 8). On termination of the application, the SP_xas well as vehicle i provide the transaction details to the RA, which acts as the arbitrator and resolves any disputes. We note that in order to lower the load of the RA, anonymous payment based protocols can be used in the LBS application access. However, we do not provide such a payment scheme here, since it is out of scope of this paper. We provide the GROUP FORMATION, GROUP JOIN, GROUP OPERATION, GROUP LEAVE and LEADER ROTATION PROTCOLS below.

3.2 Group Key and Application Address Range:

In generating the application request, vehicle i performs the following two steps: (1) randomly chooses an available address A_{aa} from a known application address range of the group G_j , (2) broadcasts the application request encrypted with the group key k_{Gj} and with Aaa as source address. The group key and the address range are obtained by the group members of G_j from GL_j, when joining the group. These two parameters prevent trace back from GL_j to i. The group key k_{Gj} prevents tracing i based on the format of application request message that is broadcast to GL_j in Step 1 of the protocol.

3.3 Protocols for Group Formation, Group Join, Group Leave, Group Operation

In the sections below, we detail the various protocols involved in the proposed location privacy scheme for VANETs.

3.3.1 Group Join Protocol:

Each vehicle (node) i, upon entering the network, periodically broadcasts safety messages for cooperative navigation. However, node i simultaneously attempts to join one of the nearest existing groups. The node i listens for broadcasts from any neighboring group leader GL_j , and then requests GL_j for membership to group G_j . A group leader can be identified by its address included in its broadcasts.

The y least significant bits of the group leader's address will be set to zero (see Group Formation protocol). GL_j verifies (using the spatial parameters of i included in the request) if i is in the range of all members of G_j. We restrict the group to have full connectivity, so that group leader rotation is possible. GL_j also verifies the public key of i included in the request, and provides i with the group key k_{Gj} and the LBS application address range, encrypted with public key of i. The pseudocode of the group join protocol is given below.

3.3.1.1 Group Join Protocol (GROUP_JOIN)

```
1. i: listen for broadcasts from neighboring group leaders H
        if (|H| > 0) and (waited for $\leq sp_{max}$)
2. i: identify $G_j \in H$ that was last heard
3. i: change $PID_{i,k-1}$ to $PID_{i,k} \in \{ PID_i \}$
4. $i \rightarrow GL_j$ :
request = $A_{GLj}$ ||$PID_{i,k-1}$|| join request
where join_request = $K_{PIDi,k-1}$ || $sign_{RA}(K_{PIDi,k-1})$
|| $location_i$ || $velocity_i$ || $acceleration_i$ ||timestamp
5. if (verified $K_{PIDi,k-1}$ ) and
(location_i is within range of node a, $\forall a \in G_j$ )
$GL_j$ : store
$PID_{i,k-1}$|| $K_{PIDi,k-1}$ || $sign_{RA}(K_{PIDi,k-1})$
$GL_j \rightarrow$ i: reply = $PID_{i,k-1}$|| $A_{GLj}$
|| $E_{KPIDi,k-1}$ ($k_{Gj}$ || app_address_range)
else
$GL_j$ : do not reply
endif
6. if (received reply within $T_{max}$)
```

```
i: set address $A_{i,j} = PID_{i,k}$
i: go to GROUP_OPERATION
else
i: identify $G_k \in H \backslash G_j$
i: set $G_j = G_k$,
if (less than $R_{max}$ repetitions without any reply)
i: go to Step 4
else
i: go to GROUP_FORM
endif
endif
else
i: go to GROUP_FORM
endif
```

3.3.2 Group Formation Protocol:

In the above protocol, the node i may not be successful in finding a group to join. The node then creates a group by means of the group formation protocol. i communicates with the RA via the RSU to obtain the group leader ID, GID_j , used in the group leader address A_{GLj} . This interaction is needed to avoid collision of the group leader addresses, since, y least significant bits of the address are set to be zero, i.e. $A_{GLj} = GID_j \parallel 0^y$. Similarly, collisions in the address range provided for LBS application access is avoided. The pseudocode for the protocol is given below.

3.3.2.1 Group Formation Protocol (GROUP_FORM)

```
if (no group heard in GROUP_JOIN) or (no group leader
replied in GROUP_JOIN)
1. i: choose $PID_{i,k} \in \{ PID_i \}$
2. i $\rightarrow$ RSU: leader_notification =
$A_{broadcast} \| PID_{i,k} \| K_{PIDi,k} \| sign_i (K_{PIDi,k})$
3. RSU,RA: verify $K_{PIDi,k}$, and generate
$E_{KPIDi,k}$
(GID$_j$ || address_range)
4. RSU $\rightarrow$ i: broadcast reply =
$PID_{i,k} \| A_{RSU} \| E_{KPIDi,k}$
(GID$_j$ || address_range)
5. i: if (received RSU reply within duration $T_{max}$)
i: generate $A_{GLj} = GID_j \| 0^y$
i: go to GROUP_OPERATION, listen for join_request
i: if (no GROUP_JOIN request) and
(waited for duration $W_{max}$)
i: go to GROUP_JOIN
else
if (number of repetitions of broadcast < $R_{max}$)
i: repeat Step 2
else
i: go to GROUP_JOIN
endif
endif
endif
```

The address_range in Step 3 is used to provide the random address A_{aa} for the anonymous access to LBS applications. We note that the address_range can directly generate A_{aa}, or alternatively, it can be used to obtain random y-bit numbers xx…x that can construct the random address $A_{aa} = GID_j \| xx\ldots x$.

3.3.3 Group Leaving Protocol:

The nodes in a VANET are highly mobile, and often a node may accelerate or change direction with time. Consequently, a node can go out of range of the group, thereby leaving its current group, and joining another group near its new location. On the other hand, a node may simply update its pseudonym/address $A_{i,j}$. In either case, the group leader GL_j of node i's current group, must assume that the node has left the group G_j. Therefore in the group leaving protocol, when GL_j does not receive any safety message broadcast with the pseudonym of node i (recorded when joining the group) for a maximum time D_{max}, GL_j assumes that either the node i has left the group or has updated its pseudonym/address $A_{i,j}$. Since in cooperative navigation, the nodes periodically broadcast navigational data with period T_n, the group leader can set the period D_{max} to be a multiple of T_n. Node i will self determine if it is out of range of GL_j, and will try to find new group by executing the group join protocol. The pseudocode for group leave protocol is as follows.

3.3.3.1 Group Leaving Protocol (GROUP_LEAVE)

```
1. i: compute current distance from group leader $GL_j$
2. i: if (going to be out of range from $GL_j$ at leave_time)
i: go to GROUP_JOIN
endif
3. $GL_j$ : if (no broadcast is received from i for duration $D_{max}$)
```

```
$GL_j$ : delete entry of $A_{i,j}$ from current group
member list
endif
```

3.3.4 Group Operation Protocol:

All the members of the group G_j participate in the group operation protocol, which consists of several subprotocols. The cooperative navigation protocol is used for safety applications. In addition, for probe data application, we include an optional probe data aggregation protocol, where the group leader aggregates the data received from the members. The aggregated data is included in the reply from the group leader to the RSU probe request in the probe data collection protocol. The group leader node cannot be provided location privacy, since it can be tracked based on its fixed pseudonym/address A_{GLj}. Hence, periodically the role of the group leader is shared by the group members. This is implemented by the leader rotation protocol. The pseudocode for the **group operation protocol** is given below, followed by the various subprotocols.

3.3.4.1 Group Operation Protocol (GROUP_OPERATION)

```
1. $G_j$ : go to COOPERATIVE_NAVIGATION
2. for all i $\in G_j \backslash GL_j$
   i: listen to broadcast sent by $GL_j$ and go to
   GROUP_LEAVE
   endfor
3. $G_j$ : optionally go to PROBE_DATA_AGGREGATION
4. $GL_j$ : go to PROBE_DATA_COLLECTION
5. if(leader rotation is needed)
   $G_j$ : go to LEADER_ROTATION
```

```
else
$GL_j$ : go to Step 3.
endif
```

In the **probe data aggregation protocol**, only a fraction of p nodes from G_j can broadcast data in each period T_d. The pseudocode for the probe data aggregation between the member of group G_j is as follows. The function aggregate_data is a suitable spatial data aggregation algorithm, and is not detailed here since it is out of the scope of this paper.

3.3.4.1.1 Probe Data Aggregation (PROBE DATA AGGREGATION)

```
1. for all i $\in G_j \backslash GL_j$
   i $\rightarrow GL_j$ : $PDATA_i = A_{GLj} \| A_{i,j} \| location_i$
   $\|$ probe_data$_i$ with probability p
   $GL_j$ : record $PDATA_i$
   endfor
2. $GL_j$ : execute aggregate_data to perform aggregation of
   all the received { $PDATA_a$ } and $PDATA_{GLj}$ , and finally
   obtain AGGREGATED_DATA
3. $G_j$ : go to Step 1 every $T_d$
```

The pseudocode for the **probe data collection protocol** is given below.

3.3.4.1.2 Probe Data Collection (PROBE_DATA_COLLECTION)

```
1. RSU $\rightarrow GL_j$ : probe_data_request = $A_{broadcast} \| A_{RSU}$
   $\|$ request_message
2. $GL_j$ : if (no AGGREGATED_DATA)
```

```
data = $location_{GLj}$ || $probe\ data_{GLj}$
else
data = $location_{GLj}$ || AGGREGATE_DATA
endif
3. $GL_j \rightarrow RSU$: reply = $A_{RSU}$ || $A_{GLj}$ || data
```

In the Step 2, the group leader checks if there is any data that was aggregated recently. If not, then it broadcasts self generated probe data. We do not specifically detail the probe_data format in this paper. Note that the probe_data_request can include specific data resolution request, i.e. for high resolution aggregated data or for lower resolution group leader only data. In the **cooperative navigation protocol**, each node independently and periodically broadcasts a safety message every T_n. In order to ensure liability of the message originator, as well as safety of the message receiver, we require each node to sign each safety message, and also include a timestamp to ensure freshness of the message. To enable verification of signature, the node includes the corresponding public key certificate. On receiving a safety message, node i verifies if the message is valid, and then performs safety computation.

3.3.4.1.3 Cooperative Navigation (COOPERATIVE_NAVIGATION)

```
1. i: $NDATA_{i,j}$ = $A_{broadcast}$ || $A_{i,j}$
|| $sign_i$(navigation $data_i$ || timestamp) || $sign_{RA}(K_{PIDi,k})$
2. for all received $NDATA_{a,x}$
i: validate and store $NDATA_{a,x}$
endfor
3. i: execute safety_computation using valid { $NDATA_{a,x}$ }
```

```
4. if(received intersection_RSU broadcast =
$A_{broadcast} || A_{IRSU} || location_{IRSU}$)
i: if (less than two replies heard)
i $\rightarrow$ intersection RSU: $A_{IRSU} || A_{i,j}$
|| navigation_data$_i$
endif
5. i: go to step 1 every $T_n$.
```

In the above protocol, the data format can be navigation_datai = (location$_i$, speed$_i$, acceleration$_i$, direction$_i$, timestamp$_i$). Steps 1-3 are used to communicate navigational data between vehicles. The Step 3 is only illustrative of the use of navigational data for safety computation. There may be other applications for such data that is not included here. The algorithm for vehicle safety computation based on the navigational data of neighboring vehicles is out of the scope of this paper. Step 4 of the protocol, is essentially used to achieve intersection vehicle collision avoidance between two groups. To avoid redundancy, not all nodes in G_j need to communicate. On the other hand, due to critical nature of the vehicle collision problem, we need to ensure protocol reliability and vehicle safety. Hence, at least two or more nodes from G_j must communicate with the RSU at the intersection. If we assume that the vehicle (on-board unit) transmission range is relatively smaller than the RSU range, the two or more nodes that reply in Step 4, will be in proximity to the intersection RSU. As mentioned earlier, in order to provide location privacy for the group leader, it becomes essential to rotate the group leader role (periodically or on demand) among the group members. The following protocol is used to enable the **rotation of the group leader** role in the group G_j .

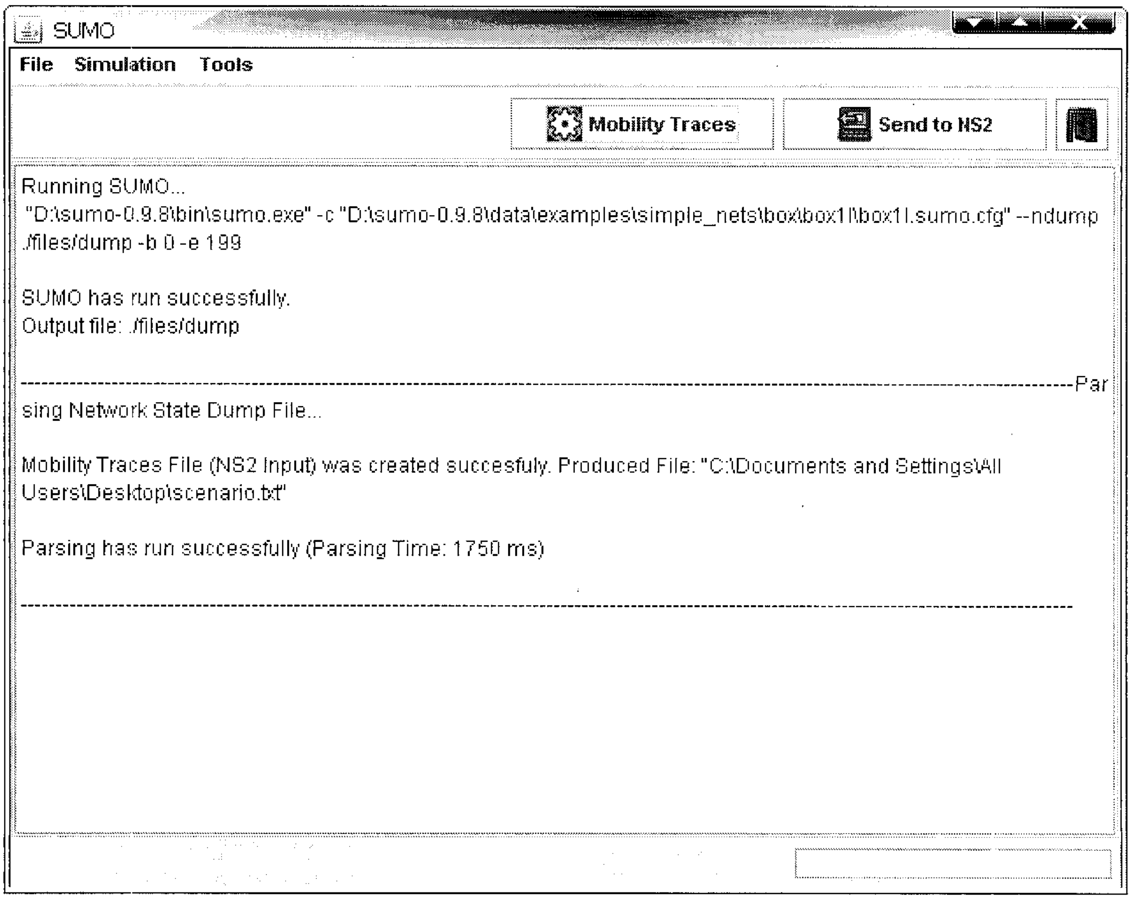
PERFORMANCE EVALUATION

4. PERFORMANCE EVALUATION:

4.1 SIMULATION ENVIRONMENT:

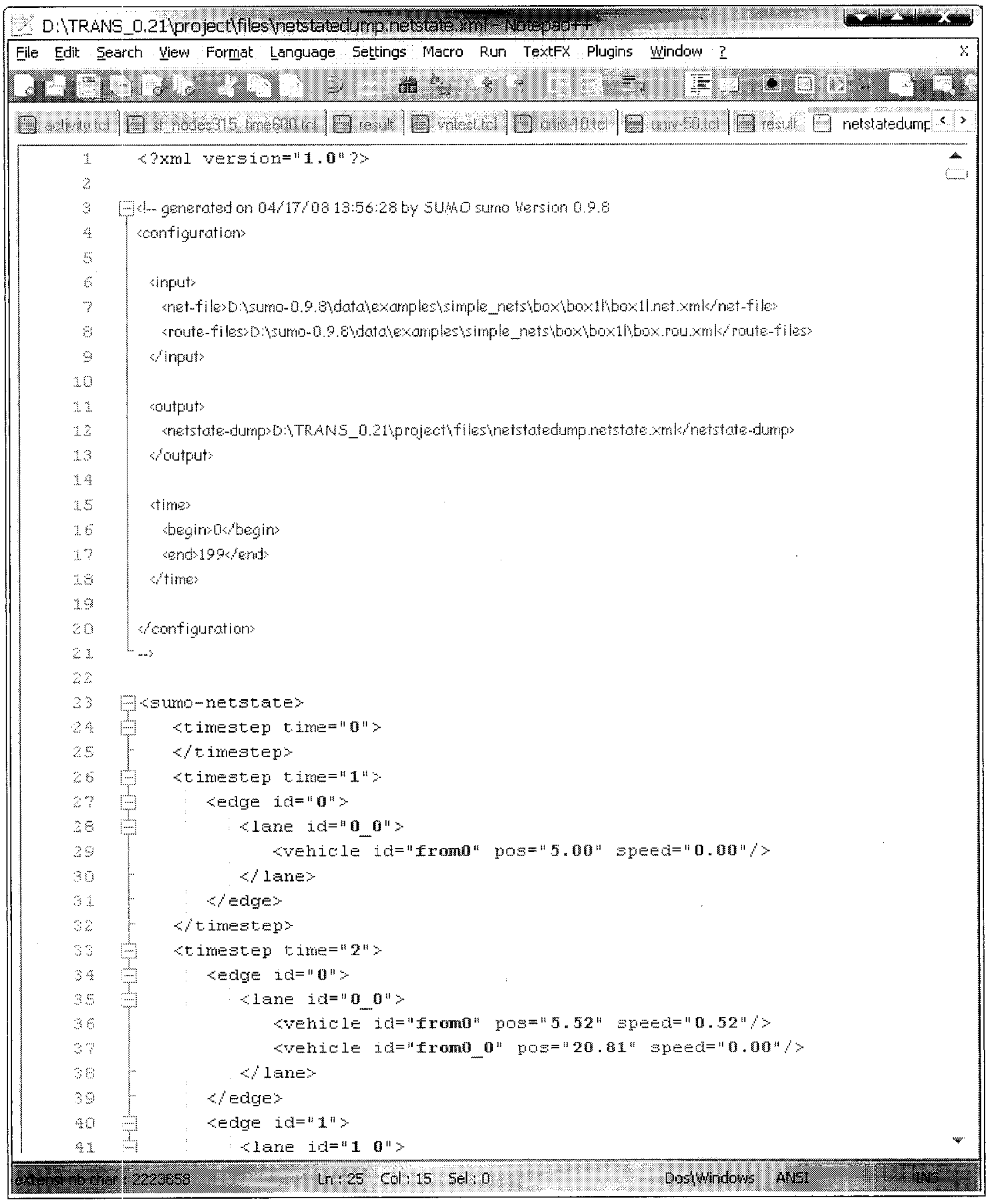
4.1.1 CONFIGURING SUMO MODULE IN TRANS:

After configuring the SUMO module, the mobility traces button is clicked.



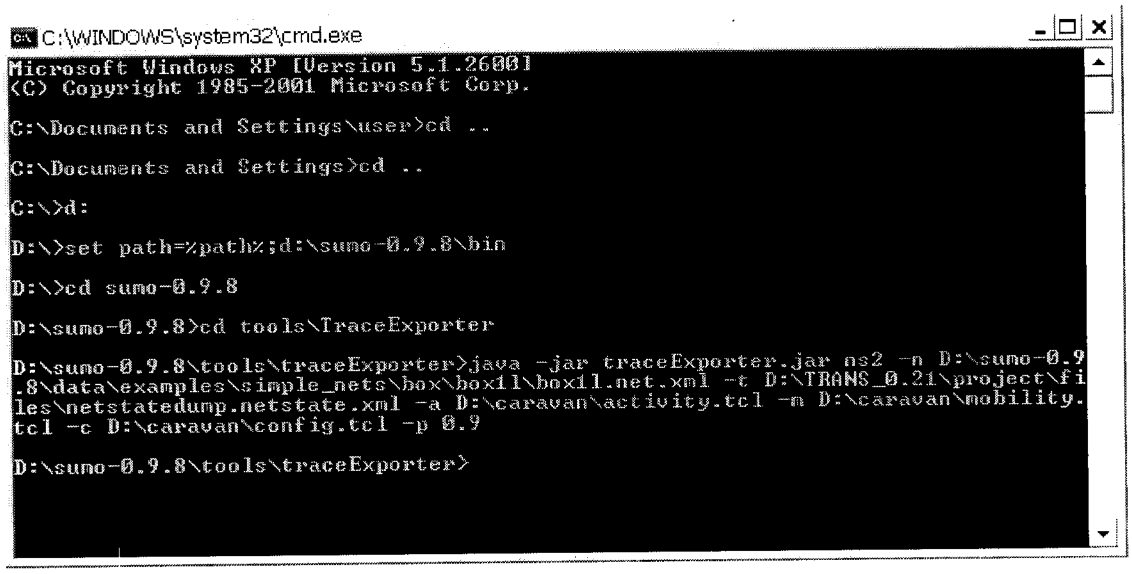
```
Running SUMO...
 "D:\sumo-0.9.8\bin\sumo.exe" -c "D:\sumo-0.9.8\data\examples\simple_nets\box\box1l\box1l.sumo.cfg" --ndump
./files/dump -b 0 -e 199

SUMO has run successfully.
Output file: ./files/dump

------------------------------------------------------------------------------------------------------------------Par
sing Network State Dump File...

Mobility Traces File (NS2 Input) was created succesfuly. Produced File: "C:\Documents and Settings\All
Users\Desktop\scenario.txt"

Parsing has run successfully (Parsing Time: 1750 ms)

---------------------------------------------------------------------------------------------------------------------
```

This produces the following dump file :


D:\TRANS_0.21\project\files\netstatedump.netstate.xml - Notepad++
File Edit Search View Format Language Settings Macro Run TextFX Plugins Window ?
activity.tcl
sf_nodes315_time600.tcl
result
vntest.tcl
univ-10.tcl
univ-50.tcl
result
netstatedump
Ln : 25 Col : 15 Sel : 0
Dos\Windows ANSI INS

```
<?xml version="1.0"?>

<!-- generated on 04/17/08 13:56:28 by SUMO sumo Version 0.9.8
<configuration>

  <input>
    <net-file>D:\sumo-0.9.8\data\examples\simple_nets\box\box1l\box1l.net.xml</net-file>
    <route-files>D:\sumo-0.9.8\data\examples\simple_nets\box\box1l\box.rou.xml</route-files>
  </input>

  <output>
    <netstate-dump>D:\TRANS_0.21\project\files\netstatedump.netstate.xml</netstate-dump>
  </output>

  <time>
    <begin>0</begin>
    <end>199</end>
  </time>

</configuration>
-->

<sumo-netstate>
   <timestep time="0">
   </timestep>
   <timestep time="1">
      <edge id="0">
         <lane id="0_0">
            <vehicle id="from0" pos="5.00" speed="0.00"/>
         </lane>
      </edge>
   </timestep>
   <timestep time="2">
      <edge id="0">
         <lane id="0_0">
            <vehicle id="from0" pos="5.52" speed="0.52"/>
            <vehicle id="from0_0" pos="20.81" speed="0.00"/>
         </lane>
      </edge>
      <edge id="1">
         <lane id="1_0">
```

Now we use a jar file called TraceExporter to generate mobility trace files, which are used as scenario files for NS2 tcl scripts.



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\user>cd ..

C:\Documents and Settings>cd ..

C:\>d:

D:\>set path=%path%;d:\sumo-0.9.8\bin

D:\>cd sumo-0.9.8

D:\sumo-0.9.8>cd tools\TraceExporter

D:\sumo-0.9.8\tools\traceExporter>java -jar traceExporter.jar ns2 -n D:\sumo-0.9
.8\data\examples\simple_nets\box\box11\box11.net.xml -t D:\TRANS_0.21\project\fi
les\netstatedump.netstate.xml -a D:\caravan\activity.tcl -m D:\caravan\mobility.
tcl -c D:\caravan\config.tcl -p 0.9

D:\sumo-0.9.8\tools\traceExporter>
```

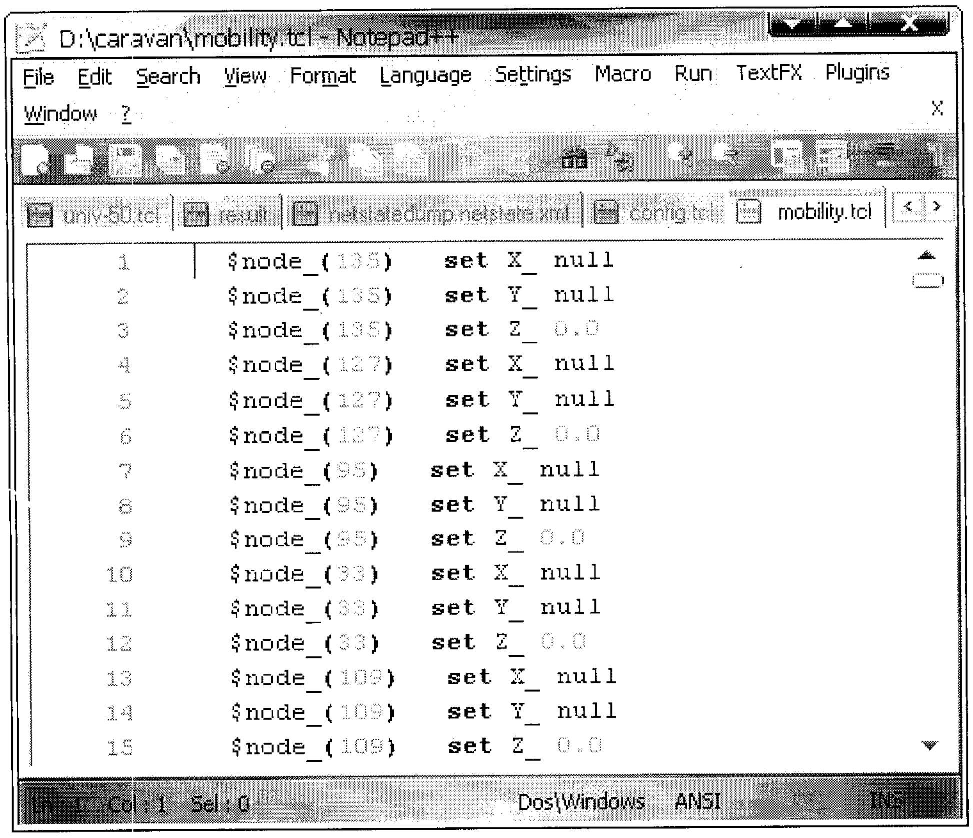
The TraceExporter generates

o Mobility file – saying when vehicles are moved and the corresponding position

o Configuration file - saying when vehicles are activated & deactivated in simulation

o Activity file – saying when vehicles will be activated and deactivated in simulation

4.1.2 MOBILITY FILE CREATED BY TRACE EXPORTER :



```
$node_(135)  set X_ null
$node_(135)  set Y_ null
$node_(135)  set Z_ 0.0
$node_(127)  set X_ null
$node_(127)  set Y_ null
$node_(127)  set Z_ 0.0
$node_(95)  set X_ null
$node_(95)  set Y_ null
$node_(95)  set Z_ 0.0
$node_(33)  set X_ null
$node_(33)  set Y_ null
$node_(33)  set Z_ 0.0
$node_(109)  set X_ null
$node_(109)  set Y_ null
$node_(109)  set Z_ 0.0
```

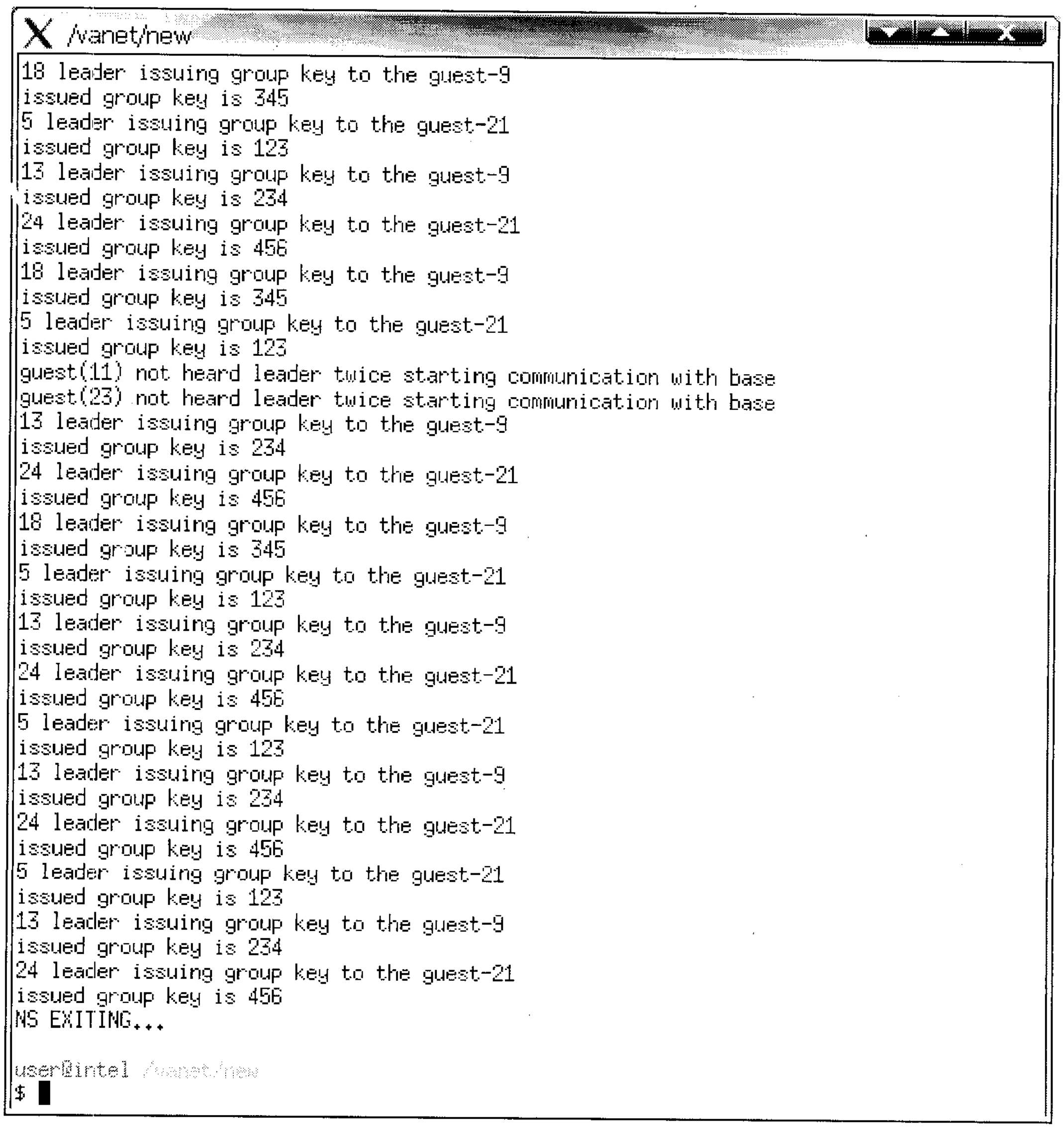
D:\caravan\mobility.tcl - Notepad++
File Edit Search View Format Language Settings Macro Run TextFX Plugins Window ?
univ-50.tcl
result
netstatedump.netstate.xml
config.tcl
mobility.tcl
Ln : 1 Col : 1 Sel : 0
Dos\Windows
ANSI
INS

This mobility file is used in the tcl scripts (NS2) as scenario files and now we run the tcl scripts using the NS2 simulator and view the animation of the result in the NAM – Network AniMator. We have used DSDV to simulate hierarchical routing between the Road Side Units and the vehicular nodes. For better results we have used the VANET RBC protocol for simulating these protocols again.

4.2 SIMULATION RESULTS :

4.2.1 SIMULATIONS USING DSDV PROTOCOL :

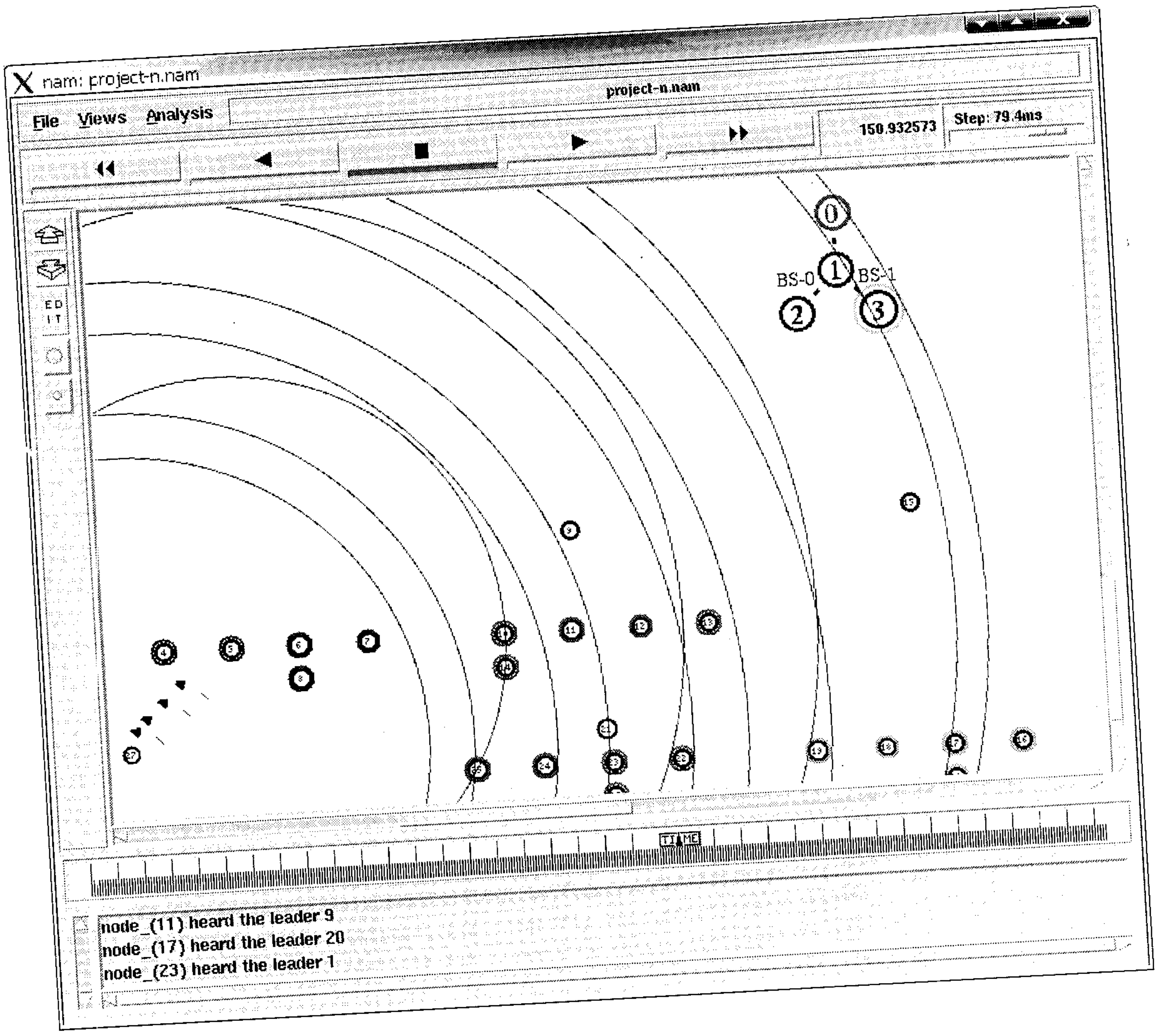
4.2.1.1 TRACE FILE FOR GROUP FORMATION & GROUP JOIN PROTCOLS:



```
18 leader issuing group key to the guest-9
issued group key is 345
5 leader issuing group key to the guest-21
issued group key is 123
13 leader issuing group key to the guest-9
issued group key is 234
24 leader issuing group key to the guest-21
issued group key is 456
18 leader issuing group key to the guest-9
issued group key is 345
5 leader issuing group key to the guest-21
issued group key is 123
guest(11) not heard leader twice starting communication with base
guest(23) not heard leader twice starting communication with base
13 leader issuing group key to the guest-9
issued group key is 234
24 leader issuing group key to the guest-21
issued group key is 456
18 leader issuing group key to the guest-9
issued group key is 345
5 leader issuing group key to the guest-21
issued group key is 123
13 leader issuing group key to the guest-9
issued group key is 234
24 leader issuing group key to the guest-21
issued group key is 456
5 leader issuing group key to the guest-21
issued group key is 123
13 leader issuing group key to the guest-9
issued group key is 234
24 leader issuing group key to the guest-21
issued group key is 456
5 leader issuing group key to the guest-21
issued group key is 123
13 leader issuing group key to the guest-9
issued group key is 234
24 leader issuing group key to the guest-21
issued group key is 456
NS EXITING...

user@intel /vanet/new
$
```

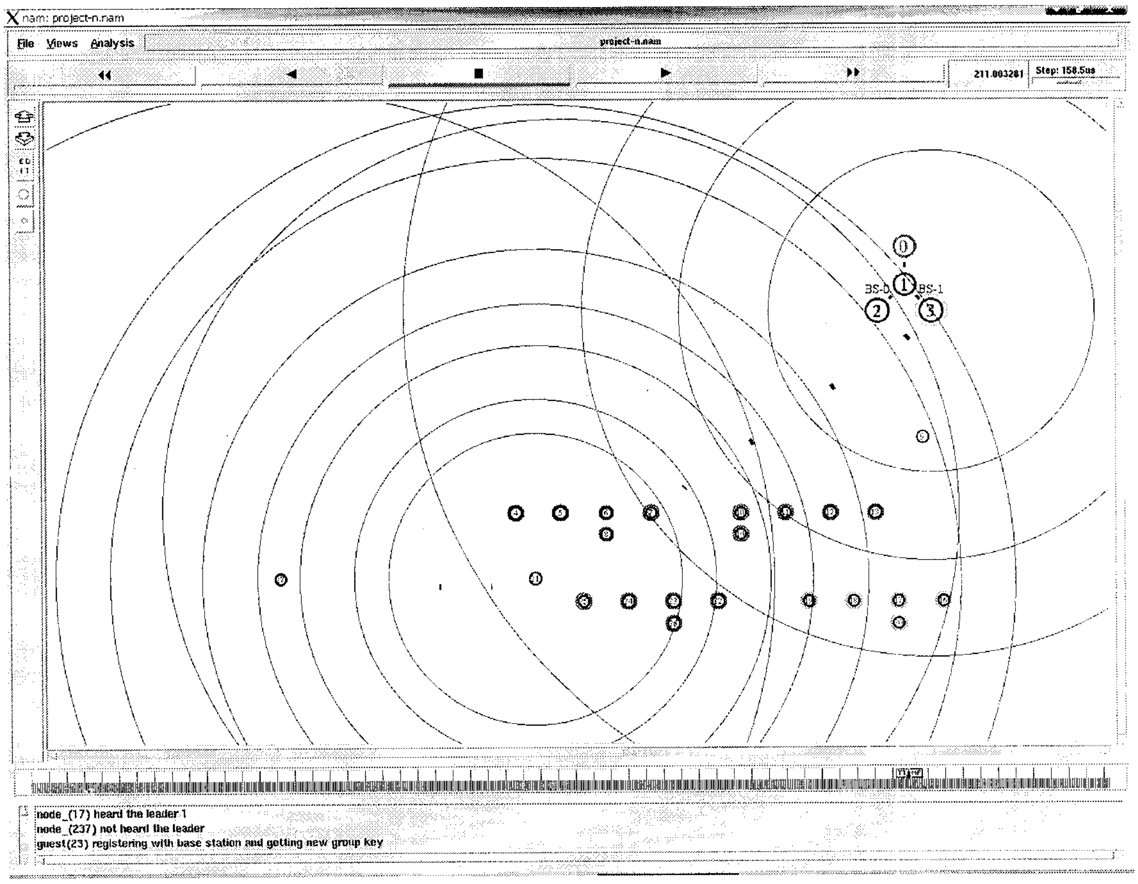
4.2.1.2 NAM OUTPUT:

4.2.1.2.1 GROUP JOIN PROTOCOL:



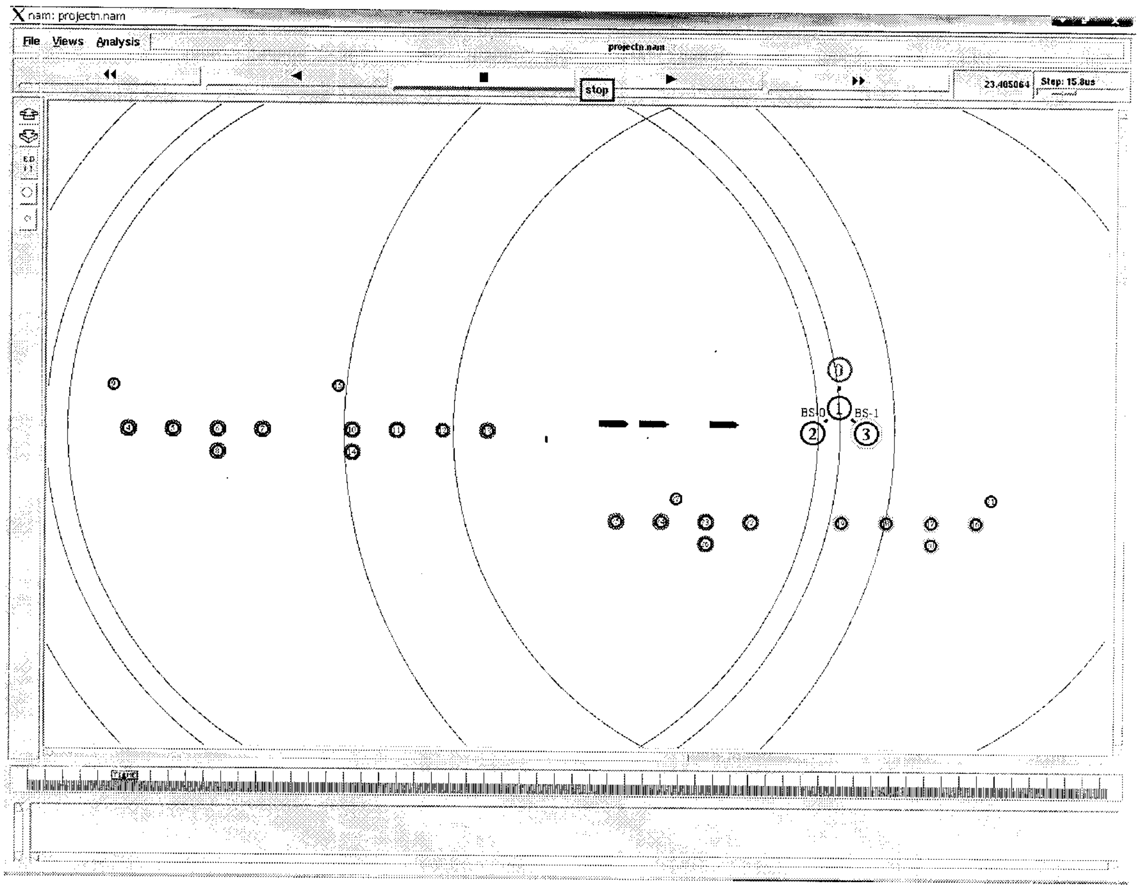
```
node_(11) heard the leader 9
node_(17) heard the leader 20
node_(23) heard the leader 1
```

4.2.1.2.2 GROUP FORMATION PROTOCOL :



```
node_(17) heard the leader 1
node_(237) not heard the leader
guest(23) registering with base station and getting new group key
```

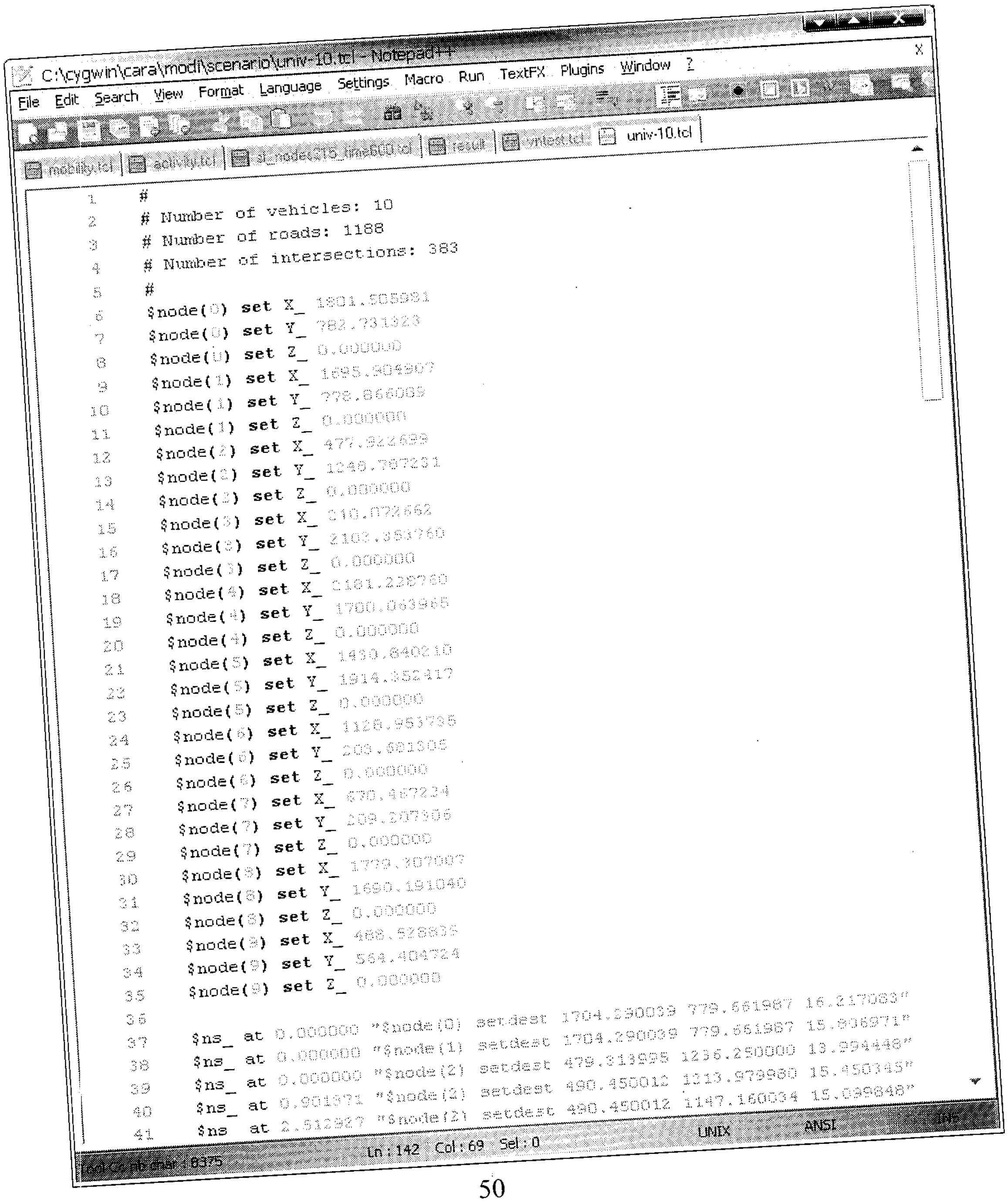
4.2.1.2.3 PROBE DATA AGGREGATION – GROUP OPERATION PROTOCOL :



4.2.2 SIMULATIONS USING VANET RBC PROTOCOL:

4.2.2.1 SIMULATION USING 10 NODES:

4.2.2.1.1 SCENARIO FILE:


C:\cygwin\cara\modi\scenario\univ-10.tcl - Notepad++
File Edit Search View Format Language Settings Macro Run TextFX Plugins Window ?
mobility.tcl
activity.tcl
result
vntest.tcl
univ-10.tcl
Ln : 142 Col : 69 Sel : 0
UNIX
ANSI
INS

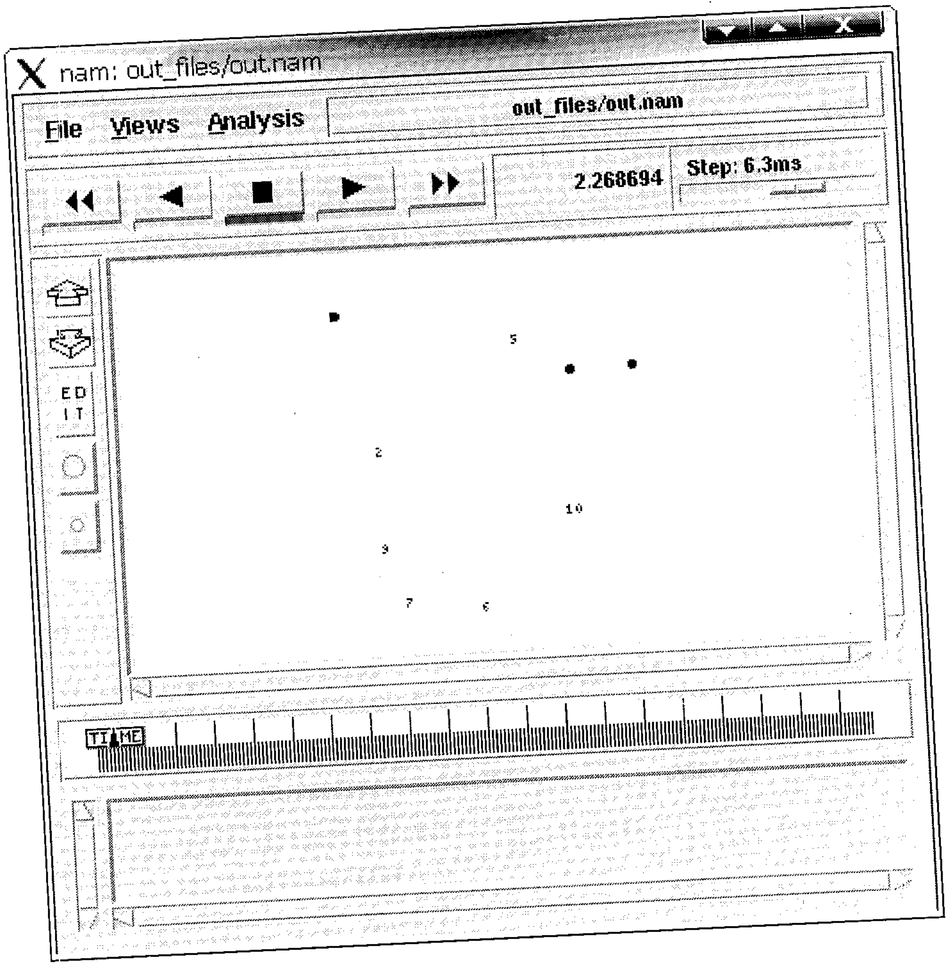
```
#
# Number of vehicles: 10
# Number of roads: 1188
# Number of intersections: 383
#
$node(0) set X_ 1801.505981
$node(0) set Y_ 782.731323
$node(0) set Z_ 0.000000
$node(1) set X_ 1695.904907
$node(1) set Y_ 778.866005
$node(1) set Z_ 0.000000
$node(2) set X_ 477.924699
$node(2) set Y_ 1240.787231
$node(2) set Z_ 0.000000
$node(3) set X_ 210.072662
$node(3) set Y_ 2103.353760
$node(3) set Z_ 0.000000
$node(4) set X_ 2101.228760
$node(4) set Y_ 1700.063965
$node(4) set Z_ 0.000000
$node(5) set X_ 1430.840210
$node(5) set Y_ 1914.352417
$node(5) set Z_ 0.000000
$node(6) set X_ 1126.953735
$node(6) set Y_ 203.601305
$node(6) set Z_ 0.000000
$node(7) set X_ 670.467234
$node(7) set Y_ 209.207306
$node(7) set Z_ 0.000000
$node(8) set X_ 1779.307007
$node(8) set Y_ 1690.191040
$node(8) set Z_ 0.000000
$node(9) set X_ 488.528835
$node(9) set Y_ 564.404724
$node(9) set Z_ 0.000000

$ns_ at 0.000000 "$node(0) setdest 1704.290039 779.661987 16.217083"
$ns_ at 0.000000 "$node(1) setdest 1704.290039 779.661987 15.806971"
$ns_ at 0.000000 "$node(2) setdest 479.313995 1236.250000 13.994448"
$ns_ at 0.901371 "$node(2) setdest 490.450012 1213.979980 15.450345"
$ns_ at 2.512927 "$node(2) setdest 490.450012 1147.160034 15.099848"
```

4.2.2.1.2 TRACE FILE:

```
num_nodes is set 10

channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5,  distCST_ = 623.3

For node 0 the key assigned is 123
0 leader issuing group key to the guest-1 time-25.000000
issued group key is 123
For node 2 the key assigned is 369
1 guest heard 0 leader time-30.000000
2 leader issuing group key to the guest-9 time-31.000000
issued group key is 369
For node 3 the key assigned is 492
1 guest heard 0 leader time-35.000000
9 guest heard 2 leader time-36.000000
For node 4 the key assigned is 615
1 guest heard 0 leader time-40.000000
9 guest heard 2 leader time-41.000000
For node 5 the key assigned is 738
1 guest heard 0 leader time-45.000000
9 guest heard 2 leader time-46.000000
4 leader issuing group key to the guest-8 time-47.000000
issued group key is 615
For node 6 the key assigned is 861
1 guest heard 0 leader time-50.000000
9 guest heard 2 leader time-51.000000
For node 7 the key assigned is 984
1 guest heard 0 leader time-55.000000
1 guest heard 0 leader time-60.000000
1 guest heard 0 leader time-65.000000
1 guest heard 0 leader time-70.000000
7 leader issuing group key to the guest-2 time-73.000000
issued group key is 984
1 guest heard 0 leader time-75.000000
0 leader issuing group key to the guest-9 time-75.000000
issued group key is 123
2 guest heard 7 leader time-78.000000
1 guest heard 0 leader time-80.000000
9 guest heard 0 leader time-80.000000
2 guest heard 7 leader time-83.000000
1 guest heard 0 leader time-85.000000
9 guest heard 0 leader time-85.000000
2 guest heard 7 leader time-88.000000
1 guest heard 0 leader time-90.000000

____________MEMBERS OF 0_____________________
```

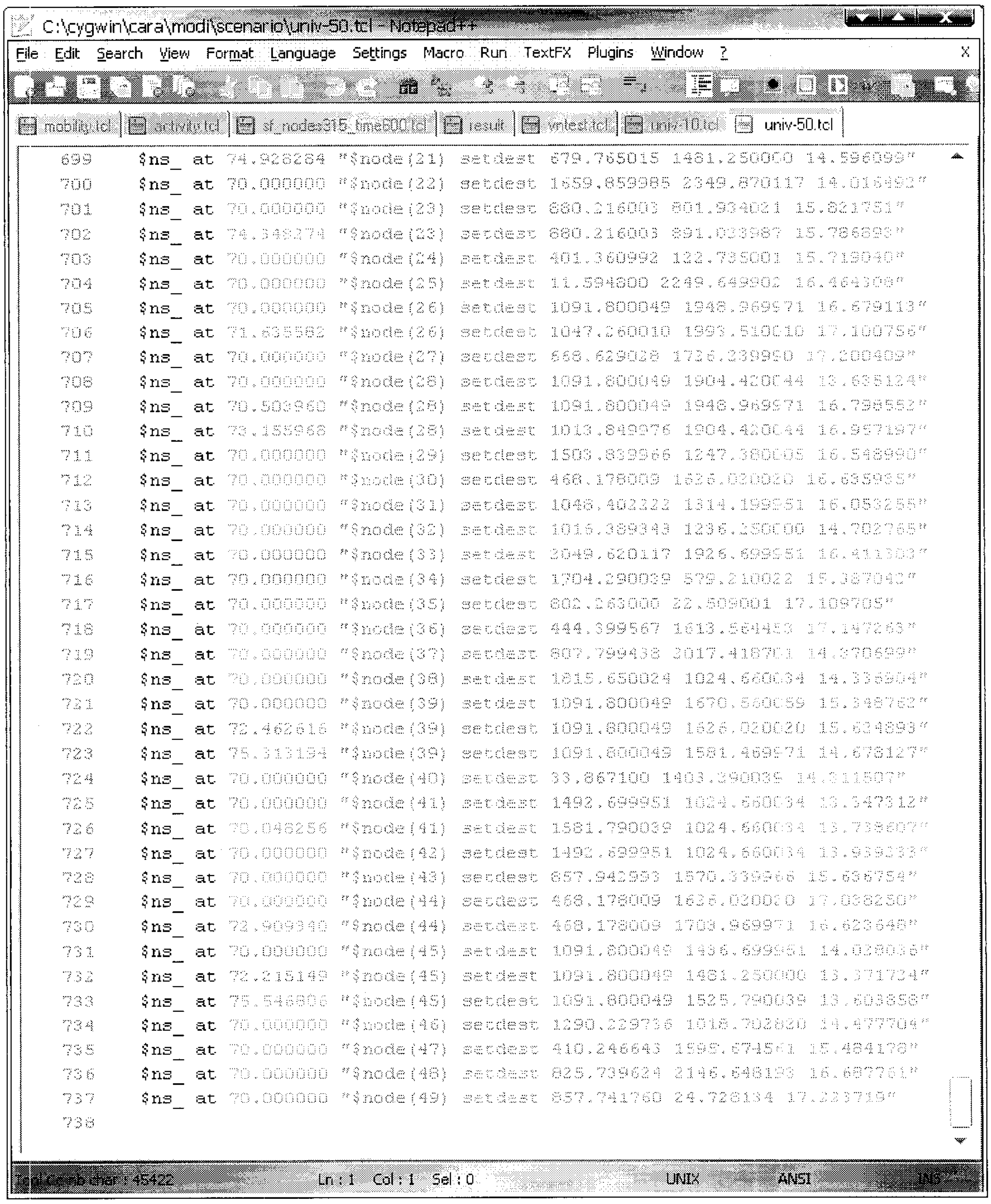
```
Group member 1 is 1
Group member 2 is 9
____________MEMBERS OF 3_____________________
____________MEMBERS OF 4_____________________
Group member 1 is 8
____________MEMBERS OF 5_____________________
____________MEMBERS OF 6_____________________
____________MEMBERS OF 7_____________________
Group member 1 is 2
```

4.2.2.1.3 NAM OUTPUT:



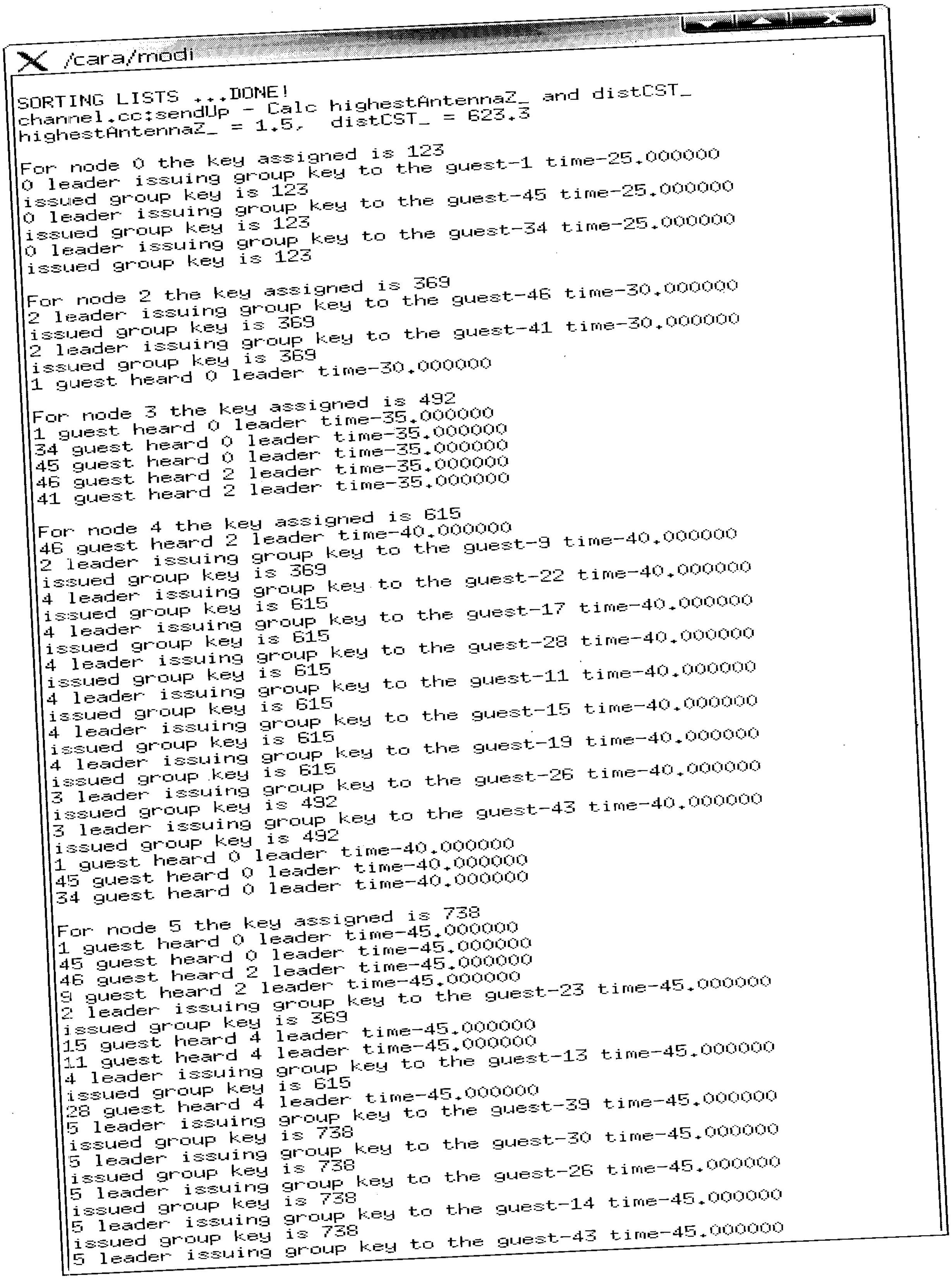
4.2.2.2 SIMULATION USING 50 NODES:

4.2.2.2.1 SCENARIO FILE :


C:\cygwin\cara\modi\scenario\univ-50.tcl - Notepad++
File Edit Search View Format Language Settings Macro Run TextFX Plugins Window ?
mobility.tcl activity.tcl sf_nodes315_time600.tcl result vntest.tcl univ-10.tcl univ-50.tcl
Total Comb char : 45422 Ln : 1 Col : 1 Sel : 0 UNIX ANSI INS

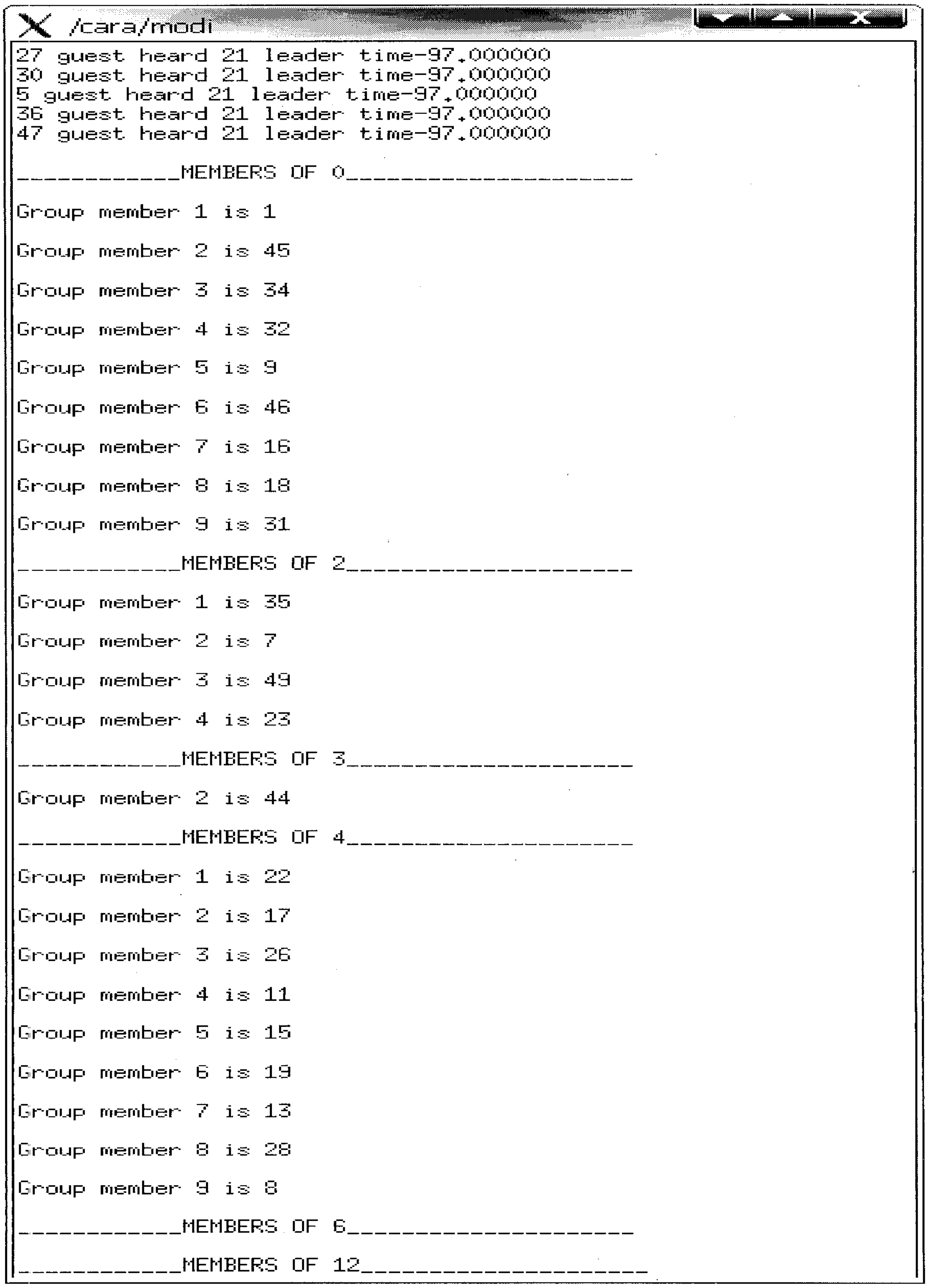
```
$ns_ at 74.926284 "$node(21) setdest 679.765015 1481.250000 14.596099"
$ns_ at 70.000000 "$node(22) setdest 1659.859985 2349.870117 14.016492"
$ns_ at 70.000000 "$node(23) setdest 680.216003 801.934021 15.821751"
$ns_ at 74.348274 "$node(23) setdest 680.216003 891.033987 15.786693"
$ns_ at 70.000000 "$node(24) setdest 401.360992 122.735001 15.719040"
$ns_ at 70.000000 "$node(25) setdest 11.594800 2249.649902 16.464308"
$ns_ at 70.000000 "$node(26) setdest 1091.800049 1948.969971 16.679113"
$ns_ at 71.635582 "$node(26) setdest 1047.260010 1993.510010 17.100756"
$ns_ at 70.000000 "$node(27) setdest 668.629028 1726.239990 17.200409"
$ns_ at 70.000000 "$node(28) setdest 1091.800049 1904.420044 13.635124"
$ns_ at 70.503960 "$node(28) setdest 1091.800049 1948.969971 16.798552"
$ns_ at 73.155968 "$node(28) setdest 1013.849976 1904.420044 16.957197"
$ns_ at 70.000000 "$node(29) setdest 1503.839966 1247.380005 16.548990"
$ns_ at 70.000000 "$node(30) setdest 468.178009 1626.020020 16.635935"
$ns_ at 70.000000 "$node(31) setdest 1048.402222 1314.199951 16.053255"
$ns_ at 70.000000 "$node(32) setdest 1016.389343 1236.250000 14.702765"
$ns_ at 70.000000 "$node(33) setdest 2049.620117 1926.699951 16.411303"
$ns_ at 70.000000 "$node(34) setdest 1704.290039 579.210022 15.387042"
$ns_ at 70.000000 "$node(35) setdest 802.263000 22.509001 17.109705"
$ns_ at 70.000000 "$node(36) setdest 444.399567 1613.564453 17.147263"
$ns_ at 70.000000 "$node(37) setdest 807.799438 2017.418701 14.370699"
$ns_ at 70.000000 "$node(38) setdest 1815.650024 1024.660034 14.336904"
$ns_ at 70.000000 "$node(39) setdest 1091.800049 1670.560059 15.348762"
$ns_ at 72.462616 "$node(39) setdest 1091.800049 1626.020020 15.634893"
$ns_ at 75.313194 "$node(39) setdest 1091.800049 1581.469971 14.678127"
$ns_ at 70.000000 "$node(40) setdest 33.867100 1403.390035 14.311507"
$ns_ at 70.000000 "$node(41) setdest 1492.699951 1024.660034 13.547312"
$ns_ at 70.048256 "$node(41) setdest 1581.790039 1024.660034 13.738607"
$ns_ at 70.000000 "$node(42) setdest 1492.699951 1024.660034 13.939333"
$ns_ at 70.000000 "$node(43) setdest 657.942993 1570.339966 15.636754"
$ns_ at 70.000000 "$node(44) setdest 468.178009 1626.020020 17.038250"
$ns_ at 72.909340 "$node(44) setdest 468.178009 1703.969971 16.623648"
$ns_ at 70.000000 "$node(45) setdest 1091.800049 1436.699951 14.038036"
$ns_ at 72.215149 "$node(45) setdest 1091.800049 1481.250000 13.371734"
$ns_ at 75.546806 "$node(45) setdest 1091.800049 1525.790039 13.603856"
$ns_ at 70.000000 "$node(46) setdest 1290.229736 1018.702820 14.477704"
$ns_ at 70.000000 "$node(47) setdest 410.246643 1599.674561 15.484178"
$ns_ at 70.000000 "$node(48) setdest 825.739624 2146.648193 16.687761"
$ns_ at 70.000000 "$node(49) setdest 857.741760 24.728134 17.223719"
```

4.2.2.2.2 TRACE FILE:



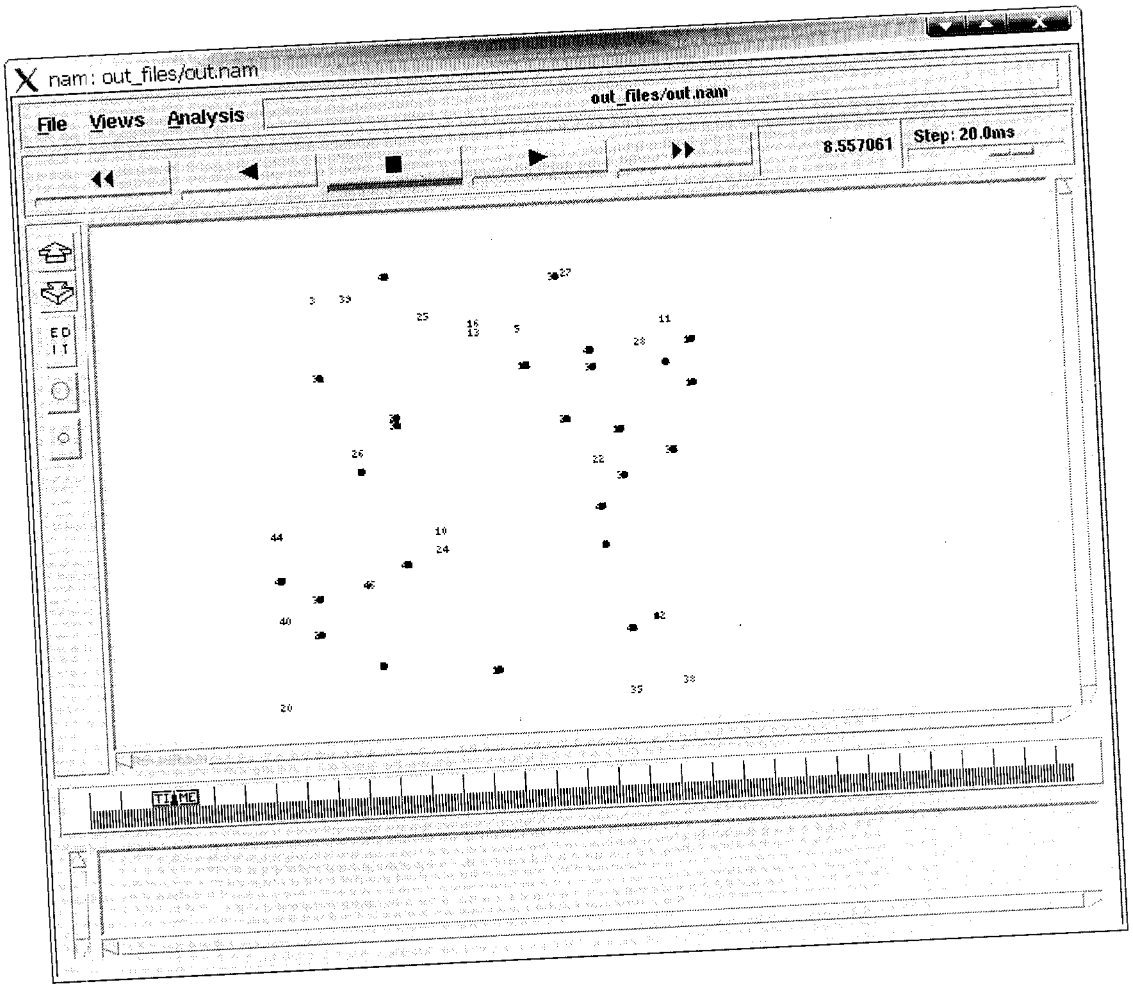
```
SORTING LISTS ...DONE!
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5,  distCST_ = 623.3

For node 0 the key assigned is 123
0 leader issuing group key to the guest-1 time-25.000000
issued group key is 123
0 leader issuing group key to the guest-45 time-25.000000
issued group key is 123
0 leader issuing group key to the guest-34 time-25.000000
issued group key is 123

For node 2 the key assigned is 369
2 leader issuing group key to the guest-46 time-30.000000
issued group key is 369
2 leader issuing group key to the guest-41 time-30.000000
issued group key is 369
1 guest heard 0 leader time-30.000000

For node 3 the key assigned is 492
1 guest heard 0 leader time-35.000000
34 guest heard 0 leader time-35.000000
45 guest heard 0 leader time-35.000000
46 guest heard 2 leader time-35.000000
41 guest heard 2 leader time-35.000000

For node 4 the key assigned is 615
46 guest heard 2 leader time-40.000000
2 leader issuing group key to the guest-9 time-40.000000
issued group key is 369
4 leader issuing group key to the guest-22 time-40.000000
issued group key is 615
4 leader issuing group key to the guest-17 time-40.000000
issued group key is 615
4 leader issuing group key to the guest-28 time-40.000000
issued group key is 615
4 leader issuing group key to the guest-11 time-40.000000
issued group key is 615
4 leader issuing group key to the guest-15 time-40.000000
issued group key is 615
4 leader issuing group key to the guest-19 time-40.000000
issued group key is 615
4 leader issuing group key to the guest-26 time-40.000000
issued group key is 615
3 leader issuing group key to the guest-43 time-40.000000
issued group key is 492
3 leader issuing group key to the guest-43 time-40.000000
issued group key is 492
1 guest heard 0 leader time-40.000000
45 guest heard 0 leader time-40.000000
34 guest heard 0 leader time-40.000000

For node 5 the key assigned is 738
1 guest heard 0 leader time-45.000000
45 guest heard 0 leader time-45.000000
46 guest heard 2 leader time-45.000000
9 guest heard 2 leader time-45.000000
2 leader issuing group key to the guest-23 time-45.000000
issued group key is 369
15 guest heard 4 leader time-45.000000
11 guest heard 4 leader time-45.000000
4 leader issuing group key to the guest-13 time-45.000000
issued group key is 615
28 guest heard 4 leader time-45.000000
5 leader issuing group key to the guest-39 time-45.000000
issued group key is 738
5 leader issuing group key to the guest-30 time-45.000000
issued group key is 738
5 leader issuing group key to the guest-26 time-45.000000
issued group key is 738
5 leader issuing group key to the guest-14 time-45.000000
issued group key is 738
5 leader issuing group key to the guest-43 time-45.000000
5 leader issuing group key to the guest-43 time-45.000000
```

4.2.2.2.2 TRACE FILE(Contd..)



```
27 guest heard 21 leader time-97.000000
30 guest heard 21 leader time-97.000000
5 guest heard 21 leader time-97.000000
36 guest heard 21 leader time-97.000000
47 guest heard 21 leader time-97.000000

____________MEMBERS OF 0_____________________

Group member 1 is 1

Group member 2 is 45

Group member 3 is 34

Group member 4 is 32

Group member 5 is 9

Group member 6 is 46

Group member 7 is 16

Group member 8 is 18

Group member 9 is 31

____________MEMBERS OF 2_____________________

Group member 1 is 35

Group member 2 is 7

Group member 3 is 49

Group member 4 is 23

____________MEMBERS OF 3_____________________

Group member 2 is 44

____________MEMBERS OF 4_____________________

Group member 1 is 22

Group member 2 is 17

Group member 3 is 26

Group member 4 is 11

Group member 5 is 15

Group member 6 is 19

Group member 7 is 13

Group member 8 is 28

Group member 9 is 8

____________MEMBERS OF 6_____________________

____________MEMBERS OF 12_____________________
```

4.2.2.2.3 NAM OUTPUT :



CONCLUSION

5. CONCLUSION:

In this project, we addressed the location privacy threats that arise in VANET due to tracking of vehicles based on their broadcasts, and proposed a solution, taking into account the mobility, and the application features in VANET. We identified that by combining neighboring vehicles into groups, it is possible to reduce the number of times a vehicle needed to broadcast for V2I applications such as probe vehicle data. Using group the vehicles can remain silent over a longer period, which in turn reduces the opportunities for the adversary to succeed in tracking the vehicle. We implemented the proposed algorithms for group join, group formation and group operation protocols using the Tool Command Language (NS2 Simulator), SUMO Simulator, TraNS and CYGWIN successfully and the results of the simulation are recorded.

FUTURE ENHANCEMENTS

6. FUTURE ENHANCEMENTS:

We have addressed the problem of location privacy by using the group concept. In this case, the Group Leader is the sole means of communication in case of LBS.The privacy of the leader is at stake since the leader is exposed to the external networks. Hence as an enhancement, the Leader Rotation Protocol can also be implemented in future, which is included in the appendix

In addition to it, since a global adversary can overhear all broadcasts, it can trace the vehicle, by relating the location of the overheard application request broadcast sent from the vehicle to the group leader, with the more frequent safety message broadcasts by the vehicle.Hence we can further enhance the project, by making the group leader to remove order of arrival information of the requests and the appearance information of the requests.

APPENDICES

7. APPENDICES :

APPENDIX 1 - Group Leader Rotation (LEADER_ROTATION)

```
1. $GL_j$ : if (do not want to be group leader) or (end of rotation
period)
$GL_j \rightarrow G_j$ : notification = $A_{broadcast} \| A_{GLj}$
$\| E_{kGj}$ { rotation_time || leader_rotation_notification }
2. forall $i \in G_j \backslash GL_j$
i: wait for random time $sp \leq sp_{max}$
i: mask y least significant bits of $PID_{i,k+1}$, and set
the masked $PID_{i,k+1}$ as $A_{GLj}new = GID_jnew$
$i \rightarrow G_j$ : reply = $A_{broadcast} \| A_{i,j}$
$\| E_{kGj}$ { leader_role_accept || $A_{GLj}new$ }
endfor
3. if ($GL_j$ receives the reply from two or more nodes in $G_j$)
$GL_j$ : choose random node i from the nodes that replied
$GL_j \rightarrow G_j$ : $A_{broadcast} \| A_{GLj}$
$\| E_{kGj}$ {leader_role_granted || $A_{i,j}$ }
else
if (no reply is received within $T_{max}$)
$GL_j$ : go to Step 1
endif
endif
4. i: broadcast leader_notification =
$A_{broadcast} \| A_{GLj}new \| PID_{i,k+1}$
5. RSU: verify leader_notification
6. RSU $\rightarrow$ i: broadcast ACK if verified to be correct
```

```
7. i: if (not received RSU ACK after waiting for $T_{max}$)
   i: repeat the broadcast in Step 4
   endif
```

Step 2-3 are used to implement the random election of the new group leader, in order to prevent any attacks that can utilize the knowledge of a deterministic election **(discussed earlier Section III-D.2).** We can further incorporate a verification mechanism in Step 3, in order to ensure the election of the new leader by the old leader node is truly
random.

APPENDIX 2 - NS2 – TCL SCRIPTS :

DSDV HEADER FILE dsdv.h :

```
#ifndef cmu_dsdv_h_
#define cmu_dsdv_h_
#include "config.h"
#include "agent.h"
#include "ip.h"
#include "delay.h"
#include "scheduler.h"
#include "queue.h"
#include "trace.h"
#include "arp.h"
#include "ll.h"
#include "mac.h"
#include "priqueue.h"
#include "rtable.h"
#if defined(WIN32) && !defined(snprintf)
#define snprintf _snprintf
```

```
#endif
typedef double Time;
#define MAX_QUEUE_LENGTH 5
#define ROUTER_PORT      0xff

class DSDV_Helper;
class DSDVTriggerHandler;
class DSDV_Agent : public Agent {
  friend class DSDV_Helper;
  friend class DSDVTriggerHandler;
public:
  DSDV_Agent();
  virtual int command(int argc, const char * const * argv);
  void lost_link(Packet *p);
  protected:
  void helper_callback(Event *e);
  Packet* rtable(int);
  virtual void recv(Packet *, Handler *);
  void trace(char* fmt, ...);
  void tracepkt(Packet *, double, int, const char *);
  void needTriggeredUpdate(rtable_ent *prte, Time t);
  void cancelTriggersBefore(Time t);
  Packet * makeUpdate(int& periodic);
   void updateRoute(rtable_ent *old_rte, rtable_ent *new_rte);
  void processUpdate (Packet * p);
  void forwardPacket (Packet * p);
  void startUp();
  int diff_subnet(int dst);
  void sendOutBCastPkt(Packet *p);
  Trace *tracetarget;       // Trace Target
  DSDV_Helper  *helper_;    // DSDV Helper, handles callbacks
  DSDVTriggerHandler *trigger_handler;
RoutingTable *table_;    // Routing Table
PriQueue *ll_queue;      // link level output queue
int seqno_;              // Sequence number to advertise with...
int myaddr_;              // My address...
char *subnet_;            // My subnet
MobileNode *node_;        // My node


char *address;
NsObject *port_dmux_;   // my port dmux
Event *periodic_callback_;          // notify for periodic update
int be_random_;
int use_mac_;
int verbose_;
int trace_wst_;
double lasttup_;                    // time of last triggered update
double next_tup;                    // time of next triggered update
double alpha_;  // 0.875
double wst0_;   // 6 (secs)
double perup_;  // 15 (secs)  period between updates
int    min_update_periods_;   // 3 we must hear an update from a neighbor
void output_rte(const char *prefix, rtable_ent *prte, DSDV_Agent *a);
};
class DSDV_Helper : public Handler {
 public:
   DSDV_Helper(DSDV_Agent *a_) { a = a_; }
   virtual void handle(Event *e) { a->helper_callback(e); }
 private:
   DSDV_Agent *a;
};
#endif
```

MODIFIED DSDV PROTOCOL :

```
extern "C" {
#include <stdarg.h>
#include <float.h>
};
#include "dsdv.h"
#include "priqueue.h"
#include <random.h>
#include <cmu-trace.h>
#include <address.h>
#include <mobilenode.h>
#define DSDV_STARTUP_JITTER 2.0
#define DSDV_ALMOST_NOW     0.1
#define DSDV_BROADCAST_JITTER 0.01
#define DSDV_MIN_TUP_PERIOD 1.0
#define IP_DEF_TTL   32
 #undef TRIGGER_UPDATE_ON_FRESH_SEQNUM
static inline double
jitter (double max, int be_random_)
{
  return (be_random_ ? Random::uniform(max) : 0);
}
void DSDV_Agent::
trace (char *fmt,...)
{
  va_list ap;
  if (!tracetarget)
   return;
  va_start (ap, fmt);
  vsprintf (tracetarget->pt_->buffer (), fmt, ap);
  tracetarget->pt_->dump ();
 va_end (ap);
}
void DSDV_Agent::tracepkt (Packet * p, double now, int me, const char *type)
{
 char buf[1024];
 unsigned char *walk = p->accessdata ();
 int ct = *(walk++);
 int seq, dst, met;
 snprintf (buf, 1024, "V%s %.5f _%d_ [%d]:", type, now, me, ct);
 while (ct--)
   {
     dst = *(walk++);
     dst = dst << 8 | *(walk++);
     dst = dst << 8 | *(walk++);
     dst = dst << 8 | *(walk++);
     met = *(walk++);
     seq = *(walk++);
     seq = seq << 8 | *(walk++);
     seq = seq << 8 | *(walk++);
     seq = seq << 8 | *(walk++);
     snprintf (buf, 1024, "%s (%d,%d,%d)", buf, dst, met, seq);
   }
 if (verbose_)
   trace ("%s", buf);
}
void DSDV_Agent::output_rte(const char *prefix, rtable_ent * prte, DSDV_Agent * a)
{
 a->trace("DFU: deimplemented");
 printf("DFU: deimplemented");
 prte = 0;
 prefix = 0;
#if 0
```

```
 printf ("%s%d %d %d %d %f %f %f %f 0x%08x\n",
          prefix, prte->dst, prte->hop, prte->metric, prte->seqnum,
          prte->udtime, prte->new_seqnum_at, prte->wst, prte->changed_at,
          (unsigned int) prte->timeout_event);
 a->trace ("VTE %.5f %d %d %d %d %f %f %f %f 0x%08x",
        Scheduler::instance ().clock (), prte->dst, prte->hop, prte->metric,
          prte->seqnum, prte->udtime, prte->new_seqnum_at, prte->wst, prte->changed_at,
           prte->timeout_event);
#endif
}
class DSDVTriggerHandler : public Handler {
 public:
  DSDVTriggerHandler(DSDV_Agent *a_) { a = a_; }
  virtual void handle(Event *e);
 private:
  DSDV_Agent *a;
};
void DSDVTriggerHandler::handle(Event *e)
{
 Scheduler & s = Scheduler::instance ();
 Time now = s.clock ();
 rtable_ent *prte;
 int update_type;        // we want periodic (=1) or triggered (=0) update?
 Time next_possible = a->lasttup_ + DSDV_MIN_TUP_PERIOD;
 for (a->table_->InitLoop(); (prte = a->table_->NextLoop());)
          if (prte->trigger_event == e) break;
 assert(prte && prte->trigger_event == e);
 if (now < next_possible)
   {
    s.schedule(a->trigger_handler, e, next_possible - now);
    a->cancelTriggersBefore(next_possible);
    return;   }
```

```
update_type = 0;
Packet * p = a->makeUpdate(/*in-out*/update_type);
if (p != NULL)
 {
   if (update_type == 1)
        {
          s.cancel(a->periodic_callback_);
          s.schedule (a->helper_, a->periodic_callback_,
          a->perup_ * (0.75 + jitter (0.25, a->be_random_)));
          if (a->verbose_) a->tracepkt (p, now, a->myaddr_, "PU");
         }
    else{
          if (a->verbose_) a->tracepkt (p, now, a->myaddr_, "TU");
         }
    assert (!HDR_CMN (p)->xmit_failure_);// DEBUG 0x2
    s.schedule (a->target_, p, jitter(DSDV_BROADCAST_JITTER, a->be_random_));
    a->lasttup_ = now; // even if we got a full update, it still counts
    }
  for (a->table_->InitLoop (); (prte = a->table_->NextLoop ());)
   if (prte->trigger_event && prte->trigger_event == e)     {
          prte->trigger_event = 0;
          delete e;
     }
 }
 void DSDV_Agent::cancelTriggersBefore(Time t)
 {
   rtable_ent *prte;
   Scheduler & s = Scheduler::instance ();
   for (table_->InitLoop (); (prte = table_->NextLoop ());)
    if (prte->trigger_event && prte->trigger_event->time_ < t)
      {
           s.cancel(prte->trigger_event);
```

```
            delete prte->trigger_event;
            prte->trigger_event = 0;
        }
}
void DSDV_Agent::needTriggeredUpdate(rtable_ent *prte, Time t)
{
  Scheduler & s = Scheduler::instance();
  Time now = Scheduler::instance().clock();
  assert(t >= now);
  if (prte->trigger_event)
    s.cancel(prte->trigger_event);
  else
    prte->trigger_event = new Event;
  s.schedule(trigger_handler, prte->trigger_event, t - now);
}
void DSDV_Agent::helper_callback (Event * e)
{
  Scheduler & s = Scheduler::instance ();
  double now = s.clock ();
  rtable_ent *prte;
  rtable_ent *pr2;
  int update_type;
  if (periodic_callback_ && e == periodic_callback_)
    {
      update_type = 1;
      Packet *p = makeUpdate(/*in-out*/update_type);
      if (verbose_)
          {
           trace ("VPC %.5f _%d_", now, myaddr_);
           tracepkt (p, now, myaddr_, "PU");
          }
      if (p) {
            assert (!HDR_CMN (p)->xmit_failure_);// DEBUG 0x2
            s.schedule (target_, p, jitter(DSDV_BROADCAST_JITTER, be_random_));
    }
        s.schedule (helper_, periodic_callback_,
                 perup_ * (0.75 + jitter (0.25, be_random_)));
    lasttup_ = now;
    return;
  }
for (table_->InitLoop (); (prte = table_->NextLoop ());)
 if (prte->timeout_event && (prte->timeout_event == e))
   break;
if (prte) {
   if (verbose_){
        trace ("VTO %.5f _%d_ %d->%d", now, myaddr_, myaddr_, prte->dst);
        }
  for (table_->InitLoop (); (pr2 = table_->NextLoop ()); ) {
        if (pr2->hop == prte->dst && pr2->metric != BIG){
           if (verbose_)
               trace ("VTO %.5f _%d_ marking %d", now, myaddr_, pr2->dst);
           pr2->metric = BIG;
           pr2->advertise_ok_at = now;
           pr2->advert_metric = true;
           pr2->advert_seqnum = true;
           pr2->seqnum++;
           needTriggeredUpdate(pr2, now);
          }}
   prte->timeout_event = 0; }
else  {
   fprintf(stderr,"DFU: unknown queue event\n");
  abort(); }
if (e)
  delete e; }
```

```
void DSDV_Agent::lost_link (Packet *p)
{
 hdr_cmn *hdrc = HDR_CMN (p);
 rtable_ent *prte = table_->GetEntry (hdrc->next_hop_);
 if(use_mac_ == 0) {
       drop(p, DROP_RTR_MAC_CALLBACK);
       return;
 }
 if (verbose_ && hdrc->addr_type_ == NS_AF_INET)
   trace("VLL %.8f %d->%d lost at %d",
   Scheduler::instance().clock(),
            hdr_ip::access(p)->saddr(), hdr_ip::access(p)->daddr(), myaddr_);
 if (!use_mac_ || !prte || hdrc->addr_type_ != NS_AF_INET)
   return;
 if (verbose_)
   trace ("VLP %.5f %d:%d->%d:%d lost at %d [hop %d]",
 Scheduler::instance ().clock (),
            hdr_ip::access (p)->saddr(),
            hdr_ip::access (p)->sport(),
            hdr_ip::access (p)->daddr(),
            hdr_ip::access (p)->dport(),
            myaddr_, prte->dst);
 if (prte->timeout_event)
   {
     Scheduler::instance ().cancel (prte->timeout_event);
     helper_callback (prte->timeout_event);
   }
 else if (prte->metric != BIG)
   {
     assert(prte->timeout_event == 0);
     prte->timeout_event = new Event ();
     helper_callback (prte->timeout_event);   }
 recv(p, 0);
#if 0
 while (p2 = ((PriQueue *) target_)->filter (prte->dst))
   {
    if (verbose_)
    trace ("VRS %.5f %d:%d->%d:%d lost at %d", Scheduler::instance ().clock (),
              hdr_ip::access (p2)->saddr(),
              hdr_ip::access (p2)->sport(),
              hdr_ip::access (p2)->daddr(),
              hdr_ip::access (p2)->dport(), myaddr_);
    recv(p2, 0);
   }
 while (p2 = ll_queue->filter (prte->dst))
   {
    if (verbose_)
    trace ("VRS %.5f %d:%d->%d:%d lost at %d", Scheduler::instance ().clock (),
              hdr_ip::access (p2)->saddr(),
              hdr_ip::access (p2)->sport(),
              hdr_ip::access (p2)->daddr(),
              hdr_ip::access (p2)->dport(), myaddr_);
    recv (p2, 0);
   }
#endif
}
static void mac_callback (Packet * p, void *arg){
 ((DSDV_Agent *) arg)->lost_link (p);
}
Packet * DSDV_Agent::makeUpdate(int& periodic){
 Packet *p = allocpkt ();
 hdr_ip *iph = hdr_ip::access(p);
 hdr_cmn *hdrc = HDR_CMN (p);
 double now = Scheduler::instance ().clock ();
```

```
rtable_ent *prte;
unsigned char *walk;
int change_count;
int rtbl_sz;
int unadvertiseable;
hdrc->next_hop_ = IP_BROADCAST;
hdrc->addr_type_ = NS_AF_INET;
iph->daddr() = IP_BROADCAST << Address::instance().nodeshift();
iph->dport() = ROUTER_PORT;
change_count = 0;
rtbl_sz = 0;
unadvertiseable = 0;
for (table_->InitLoop ();
    (prte = table_->NextLoop ()); )   {
   rtbl_sz++;
   if ((prte->advert_seqnum || prte->advert_metric)
         && prte->advertise_ok_at <= now)
        change_count++;
   if (prte->advertise_ok_at > now) unadvertiseable++;
  }
 if (change_count * 3 > rtbl_sz && change_count > 3) {      periodic = 1;
  }
 if (periodic)   {
    change_count = rtbl_sz - unadvertiseable;
    //printf("rtbsize-%d, unadvert-%d\n",rtbl_sz,unadvertiseable);
    rtable_ent rte;
    bzero(&rte, sizeof(rte));
    seqno_ += 2;
    rte.dst = myaddr_;
    rte.hop = Address::instance().get_nodeaddr(iph->saddr());
    rte.metric = 0;
    rte.seqnum = seqno_;
    rte.advertise_ok_at = 0.0; // can always advert ourselves
    rte.advert_seqnum = true;  // always include ourselves in Triggered Upds
    rte.changed_at = now;
    rte.new_seqnum_at = now;
    rte.wst = 0;
    rte.timeout_event = 0;
    rte.q = 0;
    table_->AddEntry (rte);
   }  if (change_count == 0)
   {
    Packet::free(p);
    return NULL;   }
  p->allocdata((change_count * 9) + 1);
  walk = p->accessdata ();
  *(walk++) = change_count;
  hdrc->size_ = change_count * 12 + IP_HDR_LEN;
  for (table_->InitLoop (); (prte = table_->NextLoop ());) {
    if (periodic && prte->advertise_ok_at > now) {
continue;
      }
if (periodic ||
        ((prte->advert_seqnum || prte->advert_metric)
         && prte->advertise_ok_at <= now))        {
        if (!periodic && verbose_)
          trace ("VCT %.5f _%d_ %d", now, myaddr_, prte->dst);
        *(walk++) = prte->dst >> 24;
        *(walk++) = (prte->dst >> 16) & 0xFF;
        *(walk++) = (prte->dst >> 8) & 0xFF;
        *(walk++) = (prte->dst >> 0) & 0xFF;
        *(walk++) = prte->metric;
        *(walk++) = (prte->seqnum) >> 24;
        *(walk++) = ((prte->seqnum) >> 16) & 0xFF;
        *(walk++) = ((prte->seqnum) >> 8) & 0xFF;
        *(walk++) = (prte->seqnum) & 0xFF;
        prte->last_advertised_metric = prte->metric;
        prte->advert_seqnum = false;          if (periodic)
          {          prte->advert_seqnum = false;
           prte->advert_metric = false;
          }
        change_count--;       }
  }
 assert(change_count == 0);
 return p; }
void DSDV_Agent::updateRoute(rtable_ent *old_rte, rtable_ent *new_rte){
 int negvalue = -1;
 assert(new_rte);
 Time now = Scheduler::instance().clock();
 char buf[1024];
  snprintf (buf, 1024, "%c %.5f _%d_ (%d,%d->%d,%d->%d,%d->%d,%f)",
           (new_rte->metric != BIG
            && (!old_rte || old_rte->metric != BIG)) ? 'D' : 'U',
           now, myaddr_, new_rte->dst,
           old_rte ? old_rte->metric : negvalue, new_rte->metric,
           old_rte ? old_rte->seqnum : negvalue,  new_rte->seqnum,
           old_rte ? old_rte->hop : -1,  new_rte->hop,
           new_rte->advertise_ok_at);
  table_->AddEntry (*new_rte);
  if (trace_wst_)
   trace ("VWST %.12lf frm %d to %d wst %.12lf nxthp %d [of %d]",
          now, myaddr_, new_rte->dst, new_rte->wst, new_rte->hop,
          new_rte->metric);
  if (verbose_)
   trace ("VS%s", buf);}
void DSDV_Agent::processUpdate (Packet * p){
```

```
hdr_ip *iph = HDR_IP(p);
Scheduler & s = Scheduler::instance ();
double now = s.clock ();
int i;
unsigned char *d = p->accessdata ();
unsigned char *w = d + 1;
rtable_ent rte;                // new rte learned from update being processed
rtable_ent *prte;              // ptr to entry *in* routing tbl
for (i = *d; i > 0; i--)   {
   bool trigger_update = false;  // do we need to do a triggered update?
   nsaddr_t dst;
   prte = NULL;
   dst = *(w++);
   dst = dst << 8 | *(w++);
   dst = dst << 8 | *(w++);
   dst = dst << 8 | *(w++);
   if ((prte = table_->GetEntry (dst)))        {
         bcopy(prte, &rte, sizeof(rte)); }
   else
        {
         bzero(&rte, sizeof(rte));
        }
    rte.dst = dst;
    rte.hop = Address::instance().get_nodeaddr(iph->saddr());
    rte.metric = *(w++);
    rte.seqnum = *(w++);
    rte.seqnum = rte.seqnum << 8 | *(w++);
    rte.seqnum = rte.seqnum << 8 | *(w++);
    rte.seqnum = rte.seqnum << 8 | *(w++);
    rte.changed_at = now;
    if (rte.metric != BIG) rte.metric += 1;
    if (rte.dst == myaddr_)    {
      if (rte.metric == BIG && periodic_callback_){
         s.cancel (periodic_callback_);
         s.schedule (helper_, periodic_callback_, 0);
        }
      continue;
     }
if (prte)
     { // we already have a route to this dst
      if (prte->seqnum == rte.seqnum) {
         rte.wst = prte->wst;
        rte.new_seqnum_at = prte->new_seqnum_at;
        }
      else  {
           rte.wst = alpha_ * prte->wst +
              (1.0 - alpha_) * (prte->changed_at - prte->new_seqnum_at);
        rte.new_seqnum_at = now;
        }
     }
 else { // inititallize the wst for the new route
       rte.wst = wst0_;
       rte.new_seqnum_at = now;
     }
 if (rte.metric != BIG && (!prte || prte->metric != BIG))
      rte.advertise_ok_at = now + (rte.wst * 2);
 else
      rte.advertise_ok_at = now;
 if (!prte)    {
        if (rte.metric < BIG)
          {
           rte.advert_metric = true;
           trigger_update = true;
          }
```

```
            updateRoute(prte,&rte);
          }
      else if ( prte->seqnum == rte.seqnum ) {
            if (rte.metric < prte->metric) {
                  if (rte.metric == prte->last_advertised_metric){
                    rte.advert_metric = false;
                    trigger_update = false;
                  }
              else {
                    rte.advert_metric = true;
                    trigger_update = true;
                  }
              updateRoute(prte,&rte);
             }
            else {    }
          }
      else if ( prte->seqnum < rte.seqnum )
          {  rte.advert_seqnum = true;
            if (rte.metric == prte->last_advertised_metric)
             {
              rte.advert_metric = false;
             }
            else {
                        rte.advert_metric = true;
             }
            updateRoute(prte,&rte);
#ifdef TRIGGER_UPDATE_ON_FRESH_SEQNUM
            trigger_update = true;
#else
            trigger_update = false;
#endif }
      else if ( prte->seqnum > rte.seqnum )      {
        if (rte.metric == BIG && prte->metric != BIG)
         { prte->advertise_ok_at = now;
          prte->advert_metric = true;
          needTriggeredUpdate(prte,now);
         }
        else
         { }
      }
  else
      {
       fprintf(stderr,
               "%s DFU: unhandled adding a route entry?\n", __FILE__);
       abort();
      }
  if (trigger_update)
      {
       prte = table_->GetEntry (rte.dst);
       assert(prte != NULL && prte->advertise_ok_at == rte.advertise_ok_at);
       needTriggeredUpdate(prte, prte->advertise_ok_at);
      }
  if (rte.q && rte.metric != BIG)
      {
       Packet *queued_p;
       while ((queued_p = rte.q->deque()))
        recv(queued_p, 0); }
       delete rte.q;
       rte.q = 0;
       table_->AddEntry(rte); // record the now zero'd queue
      }
 }
prte = table_->GetEntry(Address::instance().get_nodeaddr(iph->saddr()));
if (prte)   {
    if (prte->timeout_event)
        s.cancel (prte->timeout_event);
    else
        {
          prte->timeout_event = new Event ();
        }
      s.schedule (helper_, prte->timeout_event, min_update_periods_ * perup_);
   }
  else
   { bzero(&rte, sizeof(rte));
     rte.dst = Address::instance().get_nodeaddr(iph->saddr());
     rte.hop = Address::instance().get_nodeaddr(iph->saddr());
     rte.metric = 1;
     rte.seqnum = 0;
     rte.advertise_ok_at = now + 604800;
      rte.changed_at = now;
     rte.new_seqnum_at = now;
     rte.wst = wst0_;
     rte.timeout_event = new Event ();
     rte.q = 0;
         updateRoute(NULL, &rte);
      s.schedule(helper_, rte.timeout_event, min_update_periods_ * perup_);
   }
  Packet::free (p);
}
int DSDV_Agent::diff_subnet(int dst)
{
        char* dstnet = Address::instance().get_subnetaddr(dst);
        if (subnet_ != NULL) {
                if (dstnet != NULL) {
                        if (strcmp(dstnet, subnet_) != 0) {
                                delete [] dstnet;
```

```
                                    return 1;
                              }
                              delete [] dstnet;
                        }     }
        return 0;}
void DSDV_Agent::forwardPacket (Packet * p){
 hdr_ip *iph = HDR_IP(p);
 Scheduler & s = Scheduler::instance ();
 double now = s.clock ();
 hdr_cmn *hdrc = HDR_CMN (p);
 int dst;
 rtable_ent *prte;
   hdrc->direction() = hdr_cmn::DOWN;
 dst = Address::instance().get_nodeaddr(iph->daddr());
 if (diff_subnet(iph->daddr())) {
             prte = table_->GetEntry (dst);
            if (prte && prte->metric != BIG)
                  goto send;
            dst = node_->base_stn();
            prte = table_->GetEntry (dst);
            if (prte && prte->metric != BIG)
                  goto send;
 else {
                  fprintf(stderr, "warning: Route to base_stn not known: dropping pkt\n");
                  Packet::free(p);
                  return;
            }
 }
  prte = table_->GetEntry (dst);
 if (prte && prte->metric != BIG)
   {
    goto send;    }
```

```
else if (prte)
 {
           if (!prte->q)         {
        prte->q = new PacketQueue ();
      }
   prte->q->enque(p);
   if (verbose_)
       trace ("VBP %.5f _%d_ %d:%d -> %d:%d", now, myaddr_, iph->saddr(),
            iph->sport(), iph->daddr(), iph->dport());
   while (prte->q->length () > MAX_QUEUE_LENGTH)
           drop (prte->q->deque (), DROP_RTR_QFULL);
   return;
 }
else
 { // Brand new destination
   rtable_ent rte;
   double now = s.clock();
   bzero(&rte, sizeof(rte));
   rte.dst = dst;
   rte.hop = dst;
   rte.metric = BIG;
   rte.seqnum = 0;
   rte.advertise_ok_at = now + 604800;      // check back next week... :)
   rte.changed_at = now;
   rte.new_seqnum_at = now;        // was now + wst0_, why??? XXX -dam
   rte.wst = wst0_;
   rte.timeout_event = 0;
   rte.q = new PacketQueue();
   rte.q->enque(p);
   assert (rte.q->length() == 1 && 1 <= MAX_QUEUE_LENGTH);
   table_->AddEntry(rte);
   if (verbose_)
```

```
                trace ("VBP %.5f _%d_ %d:%d -> %d:%d", now, myaddr_,
                        iph->saddr(), iph->sport(), iph->daddr(), iph->dport());
    return;
  }
send:
  hdrc->addr_type_ = NS_AF_INET;
  hdrc->xmit_failure_ = mac_callback;
  hdrc->xmit_failure_data_ = this;
  if (prte->metric > 1)
          hdrc->next_hop_ = prte->hop;
  else
          hdrc->next_hop_ = dst;
  if (verbose_)
          trace ("Routing pkts outside domain: \
VFP %.5f _%d_ %d:%d -> %d:%d", now, myaddr_, iph->saddr(),
                  iph->sport(), iph->daddr(), iph->dport());

  assert (!HDR_CMN (p)->xmit_failure_ ||
          HDR_CMN (p)->xmit_failure_ == mac_callback);
  target_->recv(p, (Handler *)0);
  return;
 }
void  DSDV_Agent::sendOutBCastPkt(Packet *p)
{
  Scheduler & s = Scheduler::instance ();
  // send out bcast pkt with jitter to avoid sync
  s.schedule (target_, p, jitter(DSDV_BROADCAST_JITTER, be_random_));
}
void DSDV_Agent::recv (Packet * p, Handler *)
{
  hdr_ip *iph = HDR_IP(p);
  hdr_cmn *cmh = HDR_CMN(p);
```

```
int src = Address::instance().get_nodeaddr(iph->saddr());
int dst = cmh->next_hop();
if(src == myaddr_ && cmh->num_forwards() == 0) {
 cmh->size() += IP_HDR_LEN;
 iph->ttl_ = IP_DEF_TTL;
}
else if(src == myaddr_) {
 drop(p, DROP_RTR_ROUTE_LOOP);
 return;
}
else {
 if(--iph->ttl_ == 0) {
  drop(p, DROP_RTR_TTL);
  return;
 }
}
        Node *snode;
        Node *tnode;
      MobileNode *msnode;
        MobileNode *tmnode;
      snode =  Node::get_node_by_address(src);
        if(!(src == myaddr_))
        {
        int t = Address::instance().get_nodeaddr(myaddr_);
        tnode = Node::get_node_by_address(t);
      msnode = (MobileNode*)snode;
        tmnode = (MobileNode*)tnode;
        if(msnode->is_leader_ && tmnode->is_guest_ && !(msnode->is_guest_))
        {
        if(tmnode->leader_id_ != snode->nodeid())
        {
        printf("%d leader issuing group key to the guest-%d\n",snode->nodeid(),tnode->nodeid());
        printf("issued group key is %u\n",msnode->group_key_);
      tmnode->group_key_ = msnode->group_key_;
        tmnode->leader_id_ = snode->nodeid();
        tmnode->is_heard_ =1;
        }
        else
        printf("%d guest heard %d leader\n",tnode->nodeid(),snode->nodeid());
        }
 if ((src != myaddr_) && (iph->dport() == ROUTER_PORT))
  {
          processUpdate(p);
  }
 else if (iph->daddr() == IP_BROADCAST &&
          (iph->dport() != ROUTER_PORT))          {
           if (src == myaddr_) {
                   sendOutBCastPkt(p);
           }
           else {
                   // hand it over to the port-demux
                   port_dmux_->recv(p, (Handler*)0);
           }    }
 else    {
           forwardPacket(p);
  }
}
static class DSDVClass:public TclClass {
 public:
 DSDVClass ():TclClass ("Agent/DSDV")   { }
 TclObject *create (int, const char *const *)  {
  return (new DSDV_Agent ());
 }
} class_dsdv;
```

```
DSDV_Agent::DSDV_Agent (): Agent (PT_MESSAGE), ll_queue (0), seqno_ (0),
 myaddr_ (0), subnet_ (0), node_ (0), port_dmux_(0),
 periodic_callback_ (0), be_random_ (1),
 use_mac_ (0), verbose_ (1), trace_wst_ (0), lasttup_ (-10),
 alpha_ (0.875),  wst0_ (6), perup_ (15),
 min_update_periods_ (3)
{
 table_ = new RoutingTable ();
 helper_ = new DSDV_Helper (this);
 trigger_handler = new DSDVTriggerHandler(this);
 bind_time ("wst0_", &wst0_);
 bind_time ("perup_", &perup_);
 bind ("use_mac_", &use_mac_);
 bind ("be_random_", &be_random_);
 bind ("alpha_", &alpha_);
 bind ("min_update_periods_", &min_update_periods_);
 bind ("verbose_", &verbose_);
 bind ("trace_wst_", &trace_wst_);
  address = 0;
 }
void DSDV_Agent::startUp() {
 Time now = Scheduler::instance().clock();
  subnet_ = Address::instance().get_subnetaddr(myaddr_);
 address = Address::instance().print_nodeaddr(myaddr_);
  rtable_ent rte;
  bzero(&rte, sizeof(rte));
  rte.dst = myaddr_;
  rte.hop = myaddr_;
  rte.metric = 0;
  rte.seqnum = seqno_;
  seqno_ += 2;
  rte.advertise_ok_at = 0.0;
 rte.advert_seqnum = true;
 rte.advert_metric = true;
 rte.changed_at = now;
 rte.new_seqnum_at = now;
 rte.wst = 0;
 rte.timeout_event = 0;
   rte.q = 0;
 table_->AddEntry (rte);
periodic_callback_ = new Event ();Scheduler::instance ().schedule (helper_, periodic_callback_, jitter
DSDV_STARTUP_JITTER, be_random_));
}
int DSDV_Agent::command (int argc, const char *const *argv)
{
  if (argc == 2)
    {
      if (strcmp (argv[1], "start-dsdv") == 0)
          {
            startUp();
            return (TCL_OK);
          }
      else if (strcmp (argv[1], "dumprtab") == 0)
          {
            Packet *p2 = allocpkt ();
            hdr_ip *iph2 = HDR_IP(p2);
            rtable_ent *prte;
            printf ("Table Dump %d[%d]\n----------------------------------\n",
                     iph2->saddr(), iph2->sport());
          trace ("VTD %.5f %d:%d\n", Scheduler::instance ().clock (),
                    iph2->saddr(), iph2->sport());
Packet::free (p2);
            for (table_->InitLoop (); (prte = table_->NextLoop ());)
              output_rte ("\t", prte, this);
```

```
        printf ("\n");
        return (TCL_OK);
       }
   else if (strcasecmp (argv[1], "ll-queue") == 0)   {
       if (!(ll_queue = (PriQueue *) TclObject::lookup (argv[2])))
         {
          fprintf (stderr, "DSDV_Agent: ll-queue lookup of %s failed\n", argv[2]);
          return TCL_ERROR;
         }
        return TCL_OK;
       }
 }
else if (argc == 3)
 {
   if (strcasecmp (argv[1], "addr") == 0) {
        int temp;
        temp = Address::instance().str2addr(argv[2]);
        myaddr_ = temp;
        return TCL_OK;
   }
   TclObject *obj;
   if ((obj = TclObject::lookup (argv[2])) == 0)
       {
        fprintf (stderr, "%s: %s lookup of %s failed\n", __FILE__, argv[1],
                  argv[2]);
        return TCL_ERROR;
       }
   if (strcasecmp (argv[1], "tracetarget") == 0)
       {


        tracetarget = (Trace *) obj;
        return TCL_OK;
        }
    else if (strcasecmp (argv[1], "node") == 0) {
            node_ = (MobileNode*) obj;
            return TCL_OK;
    }
    else if (strcasecmp (argv[1], "port-dmux") == 0) {
            port_dmux_ = (NsObject *) obj;
            return TCL_OK;
    }
  }
  return (Agent::command (argc, argv));
}
```

TCL SCRIPT (DSDV IMPLEMENTATION) :

caravan.tcl

```
global opt
set opt(chan)       Channel/WirelessChannel
set opt(prop)       Propagation/TwoRayGround
set opt(netif)      Phy/WirelessPhy
set opt(mac)        Mac/802_11
set opt(ifq)        Queue/DropTail/PriQueue
set opt(ll)         LL
set opt(ant)        Antenna/OmniAntenna
set opt(x)          670
set opt(y)          670
set opt(ifqlen)     50
set opt(tr)         project-n.tr
set opt(namtr)      project-n.nam
set opt(nn)         24
set opt(adhocRouting)  DSDV
set opt(cp)         ""
set opt(sc)          ""
set opt(stop)        260
set num_wired_nodes     2
set num_bs_nodes        2
Antenna/OmniAntenna set X_ 0
Antenna/OmniAntenna set Y_ 0
Antenna/OmniAntenna set Z_ 1.5
Antenna/OmniAntenna set Gt_ 1.0
Antenna/OmniAntenna set Gr_ 1.0
Phy/WirelessPhy set CPThresh_ 10.0
Phy/WirelessPhy set CSThresh_ 1.559e-11
Phy/WirelessPhy set RXThresh_ 3.652e-10
Phy/WirelessPhy set Rb_ 2*1e6
Phy/WirelessPhy set Pt_ 0.2818
Phy/WirelessPhy set freq_ 914e+6
Phy/WirelessPhy set L_ 1.0
##########################################################
set ns_   [new Simulator]
 $ns_ node-config -addressType hierarchical
 AddrParams set domain_num_ 3
 lappend cluster_num 2 3 3
 AddrParams set cluster_num_ $cluster_num
 lappend eilastlevel 1 1 7 6 6 7 6 6
 AddrParams set nodes_num_ $eilastlevel
 set tracefd  [open $opt(tr) w]
 $ns_ trace-all $tracefd
 set namtracefd [open $opt(namtr) w]
 $ns_ namtrace-all-wireless $namtracefd 670 670
 set topo   [new Topography]
 $topo load_flatgrid $opt(x) $opt(y)
  create-god [expr $opt(nn) + $num_bs_nodes]
 set temp {0.0.0 0.1.0}
```

```
for {set i 0} {$i < $num_wired_nodes} {incr i} {
    set W($i) [$ns_ node [lindex $temp $i]]
 }
$ns_ at 0.0 "$W(0) color red"
$ns_ at 0.0 "$W(1) color blue"
 $ns_ node-config -adhocRouting $opt(adhocRouting) \
            -llType $opt(ll) \
            -macType $opt(mac) \
            -ifqType $opt(ifq) \
            -ifqLen $opt(ifqlen) \
            -antType $opt(ant) \
            -propInstance [new $opt(prop)] \
            -phyType $opt(netif) \
            -channel [new $opt(chan)] \
            -topoInstance $topo \
            -wiredRouting ON \
            -agentTrace ON \
            -routerTrace ON \
            -macTrace ON \
                -movementTrace ON \
   set temp {1.0.0 1.0.1 1.0.2 1.0.3 1.0.4 1.0.5 1.0.6 1.0.7 1.0.8 1.0.9 1.0.10 1.0.11 1.0.12 1.0.13 1.0.14
  1.0.15 1.0.16 1.0.17 1.0.18}
   set temp1 {2.0.0 2.0.1 2.0.2 2.0.3 2.0.4 2.0.5 2.0.6 2.0.7 2.0.8 2.0.9 2.0.10 2.0.11 2.0.12 2.0.13 2.0.14
  2.0.15 2.0.16 2.0.17 2.0.18}
    set BS(0) [$ns_ node [lindex $temp 0]]
    set BS(1) [$ns_ node [lindex $temp1 0]]
    $BS(0) random-motion 0
    $BS(1) random-motion 0
    $BS(0) set X_ 100.0
    $BS(0) set Y_ 150.0
    $BS(0) set Z_ 0.0
   $ns_ at 0.0 "$BS(0) label \"BS-0\""
```

```
$ns_ at 0.0 "$BS(0) add-mark m0 pink"
 $BS(1) set X_ 310.0
 $BS(1) set Y_ 150.0
 $BS(1) set Z_ 0.0
 $ns_ at 0.0 "$BS(1) label \"BS-1\""
$ns_ at 0.0 "$BS(1) add-mark m0 violet"
 $ns_ node-config -wiredRouting OFF
Phy/WirelessPhy set Pt_ 7.214e-3
 for {set j 0} {$j < 5} {incr j} {
   set node_($j) [ $ns_ node [lindex $temp \
         [expr $j+1]] ]
   $node_($j) base-station [AddrParams addr2id [$BS(0) node-addr]]
 }
Phy/WirelessPhy set Pt_ 0.2818
         set node_(5) [$ns_ node [lindex $temp 6]]
         $node_(5) base-station [AddrParams addr2id [$BS(0) node-addr]]
 $node_(0) set X_  20.0
 $node_(0) set Y_  60.0
 $node_(0) set Z_  0.0
 $node_(1) set X_  40.0
 $node_(1) set Y_  60.0
 $node_(1) set Z_  0.0
 $node_(2) set X_  60.0
 $node_(2) set Y_  60.0
 $node_(2) set Z_  0.0
 $node_(3) set X_  80.0
 $node_(3) set Y_  60.0
 $node_(3) set Z_  0.0
 $node_(4) set X_  60.0
 $node_(4) set Y_  50.0
 $node_(4) set Z_  0.0
 $node_(5) set X_  20.0
 $node_(5) set Y_  80.0
 $node_(5) set Z_  0.0
Phy/WirelessPhy set Pt_ 7.214e-3
 for {set j 6} {$j < 11} {incr j} {
   set node_($j) [ $ns_ node [lindex $temp \
         [expr $j+1]] ]
   $node_($j) base-station [AddrParams addr2id [$BS(0) node-addr]]
 }
Phy/WirelessPhy set Pt_ 0.2818
      set node_(11) [$ns_ node [lindex $temp 12]]
      $node_(11) base-station [AddrParams addr2id [$BS(0) node-addr]]
 $node_(6) set X_  120.0
 $node_(6) set Y_  60.0
 $node_(6) set Z_  0.0
 $node_(7) set X_  140.0
 $node_(7) set Y_  60.0
 $node_(7) set Z_  0.0
 $node_(8) set X_  160.0
 $node_(8) set Y_  60.0
 $node_(8) set Z_  0.0
 $node_(9) set X_  180.0
 $node_(9) set Y_  60.0
 $node_(9) set Z_  0.0
 $node_(10) set X_  120.0
 $node_(10) set Y_  50.0
 $node_(10) set Z_  0.0
 $node_(11) set X_  120.0
 $node_(11) set Y_  80.0
 $node_(11) set Z_  0.0
 #for second base station
 Phy/WirelessPhy set Pt_ 7.214e-3
   for {set j 12} {$j < 17} {incr j} {
```

```
  set node_($j) [ $ns_ node [lindex $temp1 \
        [expr $j-11]] ]
  $node_($j) base-station [AddrParams addr2id [$BS(1) node-addr]]
 }
Phy/WirelessPhy set Pt_ 0.2818
        set node_(17) [$ns_ node [lindex $temp1 6]]
        $node_(17) base-station [AddrParams addr2id [$BS(1) node-addr]]
 $node_(12) set X_  420.0
 $node_(12) set Y_  20.0
 $node_(12) set Z_  0.0
 $node_(13) set X_  400.0
 $node_(13) set Y_  20.0
 $node_(13) set Z_  0.0
 $node_(14) set X_  380.0
 $node_(14) set Y_  20.0
 $node_(14) set Z_  0.0
 $node_(15) set X_  360.0
 $node_(15) set Y_  20.0
 $node_(15) set Z_  0.0
 $node_(16) set X_  400.0
 $node_(16) set Y_  10.0
 $node_(16) set Z_  0.0
 $node_(17) set X_  420.0
 $node_(17) set Y_  30.0
 $node_(17) set Z_  0.0
 #for second base station second set
 Phy/WirelessPhy set Pt_ 7.214e-3
  for {set j 18} {$j < 23} {incr j} {
   set node_($j) [ $ns_ node [lindex $temp1 \
         [expr $j-11]] ]
   $node_($j) base-station [AddrParams addr2id [$BS(1) node-addr]]
  }
```

```
Phy/WirelessPhy set Pt_ 0.2818
        set node_(23) [$ns_ node [lindex $temp1 12]]
        $node_(23) base-station [AddrParams addr2id [$BS(1) node-addr]]
  $node_(18) set X_  320.0
  $node_(18) set Y_  20.0
  $node_(18) set Z_  0.0
  $node_(19) set X_  300.0
  $node_(19) set Y_  20.0
  $node_(19) set Z_  0.0
  $node_(20) set X_  280.0
  $node_(20) set Y_  20.0
  $node_(20) set Z_  0.0
  $node_(21) set X_  260.0
  $node_(21) set Y_  20.0
  $node_(21) set Z_  0.0
  $node_(22) set X_  300.0
  $node_(22) set Y_  10.0
  $node_(22) set Z_  0.0
  $node_(23) set X_  280.0
  $node_(23) set Y_  30.0
  $node_(23) set Z_  0.0
  $ns_ at 0.0 "$node_(0) add-mark m0 blue"
  $ns_ at 0.0 "$node_(1) add-mark m0 blue"
  $ns_ at 0.0 "$node_(2) add-mark m0 blue"
  $ns_ at 0.0 "$node_(3) add-mark m0 blue"
  $ns_ at 0.0 "$node_(4) add-mark m0 blue"
  $ns_ at 0.0 "$node_(5) add-mark m0 yellow"
  $ns_ at 0.0 "$node_(6) add-mark m0 red"
  $ns_ at 0.0 "$node_(7) add-mark m0 red"
  $ns_ at 0.0 "$node_(8) add-mark m0 red"
  $ns_ at 0.0 "$node_(9) add-mark m0 red"
  $ns_ at 0.0 "$node_(10) add-mark m0 red"
```

```
$ns_ at 0.0 "$node_(11) add-mark m0 yellow"
$ns_ at 0.0 "$node_(12) add-mark m0 green"
$ns_ at 0.0 "$node_(13) add-mark m0 green"
$ns_ at 0.0 "$node_(14) add-mark m0 green"
$ns_ at 0.0 "$node_(15) add-mark m0 green"
$ns_ at 0.0 "$node_(16) add-mark m0 green"
$ns_ at 0.0 "$node_(17) add-mark m0 yellow"
$ns_ at 0.0 "$node_(18) add-mark m0 brown"
$ns_ at 0.0 "$node_(19) add-mark m0 brown"
$ns_ at 0.0 "$node_(20) add-mark m0 brown"
$ns_ at 0.0 "$node_(21) add-mark m0 brown"
$ns_ at 0.0 "$node_(22) add-mark m0 brown"
$ns_ at 0.0 "$node_(23) add-mark m0 yellow"
$ns_ at 0.0 "$node_(2) set is_leader_ 1"
$ns_ at 0.0 "$node_(5) set is_guest_ 1"
$ns_ at 0.0 "$node_(0) set group_key_ 123"
$ns_ at 0.0 "$node_(1) set group_key_ 123"
$ns_ at 0.0 "$node_(2) set group_key_ 123"
$ns_ at 0.0 "$node_(3) set group_key_ 123"
$ns_ at 0.0 "$node_(4) set group_key_ 123"
$ns_ at 0.0 "$node_(8) set is_leader_ 1"
$ns_ at 0.0 "$node_(11) set is_guest_ 1"
$ns_ at 0.0 "$node_(6) set group_key_ 234"
$ns_ at 0.0 "$node_(7) set group_key_ 234"
$ns_ at 0.0 "$node_(8) set group_key_ 234"
$ns_ at 0.0 "$node_(9) set group_key_ 234"
$ns_ at 0.0 "$node_(10) set group_key_ 234"
$ns_ at 0.0 "$node_(14) set is_leader_ 1"
$ns_ at 0.0 "$node_(17) set is_guest_ 1"
$ns_ at 0.0 "$node_(12) set group_key_ 345"
$ns_ at 0.0 "$node_(13) set group_key_ 345"
$ns_ at 0.0 "$node_(14) set group_key_ 345"
```

```
$ns_ at 0.0 "$node_(15) set group_key_ 345"
$ns_ at 0.0 "$node_(16) set group_key_ 345"
$ns_ at 0.0 "$node_(20) set is_leader_ 1"
$ns_ at 0.0 "$node_(23) set is_guest_ 1"
$ns_ at 0.0 "$node_(18) set group_key_ 456"
$ns_ at 0.0 "$node_(19) set group_key_ 456"
$ns_ at 0.0 "$node_(20) set group_key_ 456"
$ns_ at 0.0 "$node_(21) set group_key_ 456"
$ns_ at 0.0 "$node_(22) set group_key_ 456"
#give movment to all the nodes
$ns_ at 1.0 "$node_(0) setdest 200.0 60.0 0.5"
$ns_ at 1.0 "$node_(1) setdest 220.0 60.0 0.5"
$ns_ at 1.0 "$node_(2) setdest 240.0 60.0 0.5"
$ns_ at 1.0 "$node_(3) setdest 260.0 60.0 0.5"
$ns_ at 1.0 "$node_(4) setdest 240.0 50.0 0.5"
$ns_ at 20.0 "$node_(5) setdest 440.0 100.0 1.5"
$ns_ at 1.0 "$node_(6) setdest 300.0 60.0 0.5"
$ns_ at 1.0 "$node_(7) setdest 320.0 60.0 0.5"
$ns_ at 1.0 "$node_(8) setdest 340.0 60.0 0.5"
$ns_ at 1.0 "$node_(9) setdest 360.0 60.0 0.5"
$ns_ at 1.0 "$node_(10) setdest 300.0 50.0 0.5"
$ns_ at 20.0 "$node_(11) setdest 440.0 100.0 1.5"
$ns_ at 1.0 "$node_(12) setdest 240.0 20.0 0.5"
$ns_ at 1.0 "$node_(13) setdest 220.0 20.0 0.5"
$ns_ at 1.0 "$node_(14) setdest 200.0 20.0 0.5"
$ns_ at 1.0 "$node_(18) setdest 140.0 20.0 0.5"
$ns_ at 1.0 "$node_(19) setdest 120.0 20.0 0.5"
$ns_ at 1.0 "$node_(20) setdest 100.0 20.0 0.5"
$ns_ at 1.0 "$node_(21) setdest 80.0 20.0 0.5"
$ns_ at 1.0 "$node_(22) setdest 100.0 10.0 0.5"
$ns_ at 20.0 "$node_(23) setdest 20.0 30.0 1.5"
```

```
#create links between wired and BS nodes
$ns_ duplex-link $W(0) $W(1) 5Mb 2ms DropTail
$ns_ duplex-link $W(1) $BS(0) 5Mb 2ms DropTail
$ns_ duplex-link $W(1) $BS(1) 5Mb 2ms DropTail
$ns_ duplex-link-op $W(0) $W(1) orient down
$ns_ duplex-link-op $W(1) $BS(0) orient left-down
$ns_ duplex-link-op $W(1) $BS(1) orient right-down
set tcp1 [new Agent/TCP]
$tcp1 set class_ 1
set sink1 [new Agent/TCPSink]
$ns_ attach-agent $node_(5) $tcp1
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
set tcp2 [new Agent/TCP]
$tcp2 set class_ 2
set sink2 [new Agent/TCPSink]
$ns_ attach-agent $node_(11) $tcp2
$ns_ connect $tcp2 $sink2
set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
set tcp3 [new Agent/TCP]
$tcp3 set class_ 3
set sink3 [new Agent/TCPSink]
$ns_ attach-agent $node_(17) $tcp3
#$ns_ attach-agent $node_(5) $sink3
$ns_ connect $tcp3 $sink3
set ftp3 [new Application/FTP]
$ftp3 attach-agent $tcp3
set tcp4 [new Agent/TCP]
$tcp4 set class_ 4
set sink4 [new Agent/TCPSink]
$ns_ attach-agent $node_(23) $tcp4
```

```
 $ns_ connect $tcp4 $sink4
 set ftp4 [new Application/FTP]
 $ftp4 attach-agent $tcp4
 for {set i 0} {$i < $opt(nn)} {incr i} {
    $ns_ initial_node_pos $node_($i) 5
  }
$ns_ at 50.0 "$node_(2) set is_leader_ 0"
$ns_ at 50.0 "$node_(1) set is_leader_ 1"
$ns_ at 70.0 "$node_(8) set is_leader_ 0"
$ns_ at 70.0 "$node_(9) set is_leader_ 1"
set cnt5 0
set cnt11 0
set cnt17 0
set cnt23 0
proc check {} {
        global ns_ node_ ftp1 ftp2 ftp3 ftp4 cnt5 cnt11 cnt17 cnt23 sink1 sink2 sink3 sink4 tcp1 tcp2 tcp3
tcp4
        global BS
        set time 15
        set now [$ns_ now]
        set hd5 [$node_(5) set is_heard_]
        set ld5 [$node_(5) set is_leader_]
        if {$ld5 == 1} {
        $ns_ at $now "$ns_ trace-annotate \"node_(5) has become leader\""
        } elseif {$hd5 == 1} {
        set $hd5 0
        set $cnt5 0
        set lr_id [$node_(5) set leader_id_]
        set lr_id [expr $lr_id - 4]
        $ns_ at $now "$ns_ trace-annotate \"node_(5) heard the leader $lr_id\""
        $ns_ at $now "$ns_ attach-agent $node_($lr_id) $sink1"
        $ns_ at $now "$ns_ connect $tcp1 $sink1"
        $ns_ at $now "$ftp1 start"
        $ns_ at [expr $now + 0.2] "$ftp1 stop"
        $ns_ at $now "$node_(5) set is_heard_ 0"
        } else {
        set cnt5 [expr $cnt5 + 1]
      $ns_ at $now "$ns_ trace-annotate \"node_(5) not heard the leader\""
        }
        if {$cnt5 == 2} {
      puts "guest(5) not heard leader twice starting communication with base to become leader"
      $ns_ at$now"$ns_ trace-annotate\"guest(5) registering with base station& getting new group key\""
      $ns_ at $now "$ns_ attach-agent $BS(0) $sink1"
      $ns_ at $now "$ftp1 start"
      $ns_  at [expr $now + 2.0 ] "$ftp1 stop"
      $ns_ at $now "$node_(5) set is_leader_ 1"
      set cnt5 0
         }
set hd11 [$node_(11) set is_heard_]
set ld11 [$node_(11) set is_leader_]
if { $ld11 == 1} {
         $ns_ at $now "$ns_ trace-annotate \"node_(11) has become leader\""
         } elseif {$hd11 == 1} {
set $hd11 0
set $cnt11 0
set lr_id [$node_(11) set leader_id_]
set lr_id [expr $lr_id - 4]
       $ns_ at $now "$ns_ trace-annotate \"node_(11) heard the leader $lr_id\""
       $ns_ at $now "$ns_ attach-agent $node_($lr_id) $sink2"
       $ns_ at $now "$ns_ connect $tcp2 $sink2"
       $ns_ at $now "$ftp2 start"
       $ns_ at [expr $now + 0.2] "$ftp2 stop"
       $ns_ at $now "$node_(11) set is_heard_ 0"
       } else {
```

```
set cnt11 [expr $cnt11 + 1]
$ns_ at $now "$ns_ trace-annotate \"node_(11) not heard the leader\""
}
if {$cnt11 == 2} {
puts "guest(11) not heard leader twice starting communication with base to become leader"
$ns_at$now"$ns_trace-annotate\"guest(11)registering with base station & getting new group key\""
$ns_ at $now "$ns_ attach-agent $BS(0) $sink2"
$ns_ at $now "$ftp2 start"
$ns_  at [expr $now + 2.0 ] "$ftp2 stop"
$ns_ at $now "$node_(11) set is_leader_ 1"
set cnt11 0
}
set hd17 [$node_(17) set is_heard_]
set ld17 [$node_(17) set is_leader_]
if { $ld17 == 1} {
$ns_ at $now "$ns_ trace-annotate \"node_(17) has become leader\""
} elseif {$hd17 == 1} {
set $hd17 0
set $cnt17 0
set lr_id [$node_(17) set leader_id_]
set lr_id [expr $lr_id - 4]
$ns_ at $now "$ns_ trace-annotate \"node_(17) heard the leader $lr_id\""
$ns_ at $now "$ns_ attach-agent $node_($lr_id) $sink3"
$ns_ at $now "$ns_ connect $tcp3 $sink3"
$ns_ at $now "$ftp3 start"
$ns_ at [expr $now + 0.2] "$ftp3 stop"
$ns_ at $now "$node_(17) set is_heard_ 0"
} else {
set cnt17 [expr $cnt17 + 1]
$ns_ at $now "$ns_ trace-annotate \"node_(17) not heard the leader\""        }
if {$cnt17 == 2} {
puts "guest(17) not heard leader twice starting communication with base to become leader"
```

```
$ns_ at $now "$ns_ trace-annotate \"guest(17) registering with base station and getting new group
key\""
$ns_ at $now "$ns_ attach-agent $BS(1) $sink3"
$ns_ at $now "$ftp3 start"
$ns_ at [expr $now + 2.0 ] "$ftp3 stop"
$ns_ at $now "$node_(17) set is_leader_ 1"
set cnt17 0      }
set hd23 [$node_(23) set is_heard_]
set ld23 [$node_(23) set is_leader_]
if { $ld23 == 1} {
$ns_ at $now "$ns_ trace-annotate \"node_(23) has become leader\""
} elseif {$hd23 == 1} {
set $hd23 0
set $cnt23 0
set lr_id [$node_(23) set leader_id_]
set lr_id [expr $lr_id - 4]
$ns_ at $now "$ns_ trace-annotate \"node_(23) heard the leader $lr_id\""
$ns_ at $now "$ns_ attach-agent $node_($lr_id) $sink4"
$ns_ at $now "$ns_ connect $tcp4 $sink4"
$ns_ at $now "$ftp4 start"
$ns_ at [expr $now + 1.0] "$ftp4 stop"
$ns_ at $now "$node_(23) set is_heard_ 0"
} else {
set cnt23 [expr $cnt23 + 1]
$ns_ at $now "$ns_ trace-annotate \"node_(237) not heard the leader\""
}
if {$cnt23 == 2} {
puts "guest(23) not heard leader twice starting communication with base to become leader"
$ns_ at $now "$ns_ trace-annotate \"guest(23) registering with base station and getting new group
key\""
$ns_ at $now "$ns_ attach-agent $BS(1) $sink4"
$ns_ at $now "$ftp4 start"
```

```
      $ns_  at [expr $now + 2.0 ] "$ftp4 stop"
      $ns_ at $now "$node_(23) set is_leader_ 1"
      set cnt23 0
      }
$ns_ at [expr $now+$time] "check"
}
$ns_ at 15.0 "check"
 for {set i } {$i < $opt(nn) } {incr i} {
     $ns_ at $opt(stop).0000010 "$node_($i) reset";
  }
  $ns_ at $opt(stop).0000010 "$BS(0) reset";
  $ns_ at $opt(stop).0000010 "$BS(1) reset";
  $ns_ at $opt(stop).1 "puts \"NS EXITING...\" ; $ns_ halt"
  puts "Starting Simulation..."
  $ns_ run
```

MODIFIED MOBILE NODE TO SUIT VANET RBC PROTOCOL :

```
#include <math.h>
#include <stdlib.h>
#include "connector.h"
#include "delay.h"
#include "packet.h"
#include "random.h"
#include "trace.h"
#include "address.h"
#include "arp.h"
#include "topography.h"
#include "ll.h"
#include "mac.h"
#include "propagation.h"
#include "mobilenode.h"
```

```
#include "phy.h"
#include "wired-phy.h"
#include "god.h"
static LIST_HEAD(_dummy_MobileNodeList, MobileNode) nodehead = { 0 };
static class MobileNodeClass : public TclClass {
public:
      MobileNodeClass() : TclClass("Node/MobileNode") {}
      TclObject* create(int, const char*const*) {
            return (new MobileNode);
      }
} class_mobilenode;
void PositionHandler::handle(Event*){
         Scheduler& s = Scheduler::instance();
#if 0
         fprintf(stderr, "*** POSITION HANDLER for node %d (time: %f) ***\n",
                  node->address(), s.clock());
#endif
         node->update_position();
#ifdef DEBUG
       fprintf(stderr, "%d - %s: calling random_destination()\n",
             node->address_, __PRETTY_FUNCTION__);
#endif
         node->random_destination();
         s.schedule(&node->pos_handle_, &node->pos_intr_,
                     node->position_update_interval_);}
MobileNode::MobileNode(void) : pos_handle_(this){
         X_ = Y_ = Z_ = speed_ = 0.0;
         dX_ = dY_ = dZ_ = 0.0;
         destX_ = destY_ = 0.0;
         random_motion_ = 0;
         base_stn_ = -1;
         T_ = 0;
```

```
        log_target_ = 0;
        next_ = 0;
        radius_ = 0;
        position_update_interval_ = POSITION_UPDATE_INTERVAL;
        position_update_time_ = 0.0;
        is_leader_=group_key_=is_guest_=is_heard_=0;
        for(int i=0;i<50;i++)
        group[i]=-1;
        LIST_INSERT_HEAD(&nodehead, this, link_);       // node list
        LIST_INIT(&ifhead_);                            // interface list
        bind("X_", &X_);
        bind("Y_", &Y_);
        bind("Z_", &Z_);
        bind("speed_", &speed_);
        bind("is_leader_",&is_leader_);
        bind("group_key_",&group_key_);
        bind("is_guest_",&is_guest_);
        bind("is_heard_",&is_heard_);
        bind("leader_id_",&leader_id_);
        bind("count_",&count_);
        for(int i=0;i<50;i++)  {
        char group[100];
        sprintf(group, "group[%d]",i);
        int n=group[i];
        bind("group",&n);
        }}
int MobileNode::command(int argc, const char*const* argv){
        Tcl& tcl = Tcl::instance();
        if(argc == 2) {
                if(strcmp(argv[1], "start") == 0) {
                        start();
                          return TCL_OK;       }}}
```

VANET RBC HEADER :

```
#ifndef vanetrbc_h
#define vanetrbc_h
#include <agent.h>
#include <mobilenode.h>
#include <tclcl.h>
#include <packet.h>
#include <address.h>
#include <ip.h>
#include <timer-handler.h>
#include <rng.h>
#include <trace.h>
#include "vanetrbc_header.h"
#include "vanetrbc_rxdatadb.h"
#define JITTER (rng_.uniform(0, 1))
class VanetRBCAgent; // forward declaration
class VanetRBC_PktTimer : public TimerHandler {
protected:
   VanetRBCAgent* agent_;
   virtual void expire(Event* e);
public:
   VanetRBC_PktTimer(VanetRBCAgent* agent) : TimerHandler() {
      agent_ = agent;
   }
};
class VanetRBCAgent : public Agent {
   friend class VanetRBC_PktTimer;
public:
   VanetRBCAgent(u_int32_t);
   virtual int command(int argc, const char*const* argv);
   virtual void recv(Packet*, Handler*);
```

```
  virtual void timeout(int);
protected:
MobileNode* node_;
        int group[50];
  VanetRBC_PktTimer pkt_timer_;  // timer for sending packets
  double interval_;              // sending interval
  bool running_;                 // periodic sending
  u_int32_t vanetID_;            // own ID
  vanetrbc_rxdatadb rxdb_;       // received packets database
  Trace* logtarget_;             // for logging
  RNG rng_;                      // random number generator, for the jitter
  double jitterfactor_;          // multiplication-factor for the tx-jitter
  double crypto_delay_;          // signing and verification delay [s]
  void sendRBC_pkt();            // broadcast a RBC message
  void recvRBC(Packet*);         // processing of received RBC messages
  void printgroup(u_int32_t);
  void reset_vanetrbc_pkt_timer();  // for the timer
};
#endif
```

VANET RBC PROTOCOL :

```
#include <assert.h>
#include "vanetrbc.h"
int hdr_vanet::offset_;
static class VanetRBCHeaderClass : public PacketHeaderClass {
public:
  VanetRBCHeaderClass() : PacketHeaderClass("PacketHeader/VanetRBC",
                                   sizeof(hdr_all_vanet)) {
    bind_offset(&hdr_vanet::offset_);
  }
} class_vanetrbchdr;
```

```
static class VanetRBCClass : public TclClass {
public:
  VanetRBCClass() : TclClass("Agent/VanetRBC") { };
  TclObject* create(int argc, const char*const* argv) {
    if(argc != 5) {
      printf("Creating VanetRBC-Agent: argc not equal 5: %d\n", argc);
      exit(1);
    }
    return (new VanetRBCAgent(atoi(argv[4])));
  }
} class_vanetrbc;
void VanetRBCAgent::reset_vanetrbc_pkt_timer() {
  pkt_timer_.resched((double)interval_);
}
void VanetRBC_PktTimer::expire(Event*) {
  agent_->timeout(0);
}
VanetRBCAgent::VanetRBCAgent(u_int32_t id) : Agent(PT_VANETRBC),  pkt_timer_(this) {
  vanetID_ = id;
  running_ = false;
  bind("jitterFactor_", &jitterfactor_);
  bind("crypto_delay_", &crypto_delay_);
  bind_time("interval_", &interval_);
  node_ = (MobileNode*)Node::get_node_by_address(id);
  long int rngseed = 0;
  rng_.set_seed(rngseed); // initialize RNG (for JITTER)
}
int VanetRBCAgent::command(int argc, const char*const* argv) {
  if (argc == 2) {     // a command without additional parameter
    if (logtarget_ != 0) {
      if (strcmp(argv[1], "start-regbc") == 0) {
        running_ = true;   // we want the timer to be rescheduled
        sendRBC_pkt();      // send the first message
        return (TCL_OK);   // always return OK if we get until here
      }
      else if (strcmp(argv[1], "stop-regbc") == 0) {
running_ = false;       // stop the timers
        pkt_timer_.cancel();

        return (TCL_OK);
      }
            else if (strcmp(argv[1], "assignkey") == 0) {
                  MobileNode *tnode =MobileNode*)Node::get_node_by_address(vanetID_);
                  tnode->group_key_ = (123*vanetID_)+123;
                  tnode->is_leader_ = 1;
                  tnode->count_ = 0;
                  printf("\nFor node %d the key assigned is %d\n",vanetID_,tnode->group_key_);
                  return TCL_OK;
                  }
      else if (strcmp(argv[1], "dump-rxdb") == 0) {
        sprintf(logtarget_->pt_->buffer(),
                  "Node %u: RxDB (t: %f)", vanetID_,
                      Scheduler::instance().clock());
        logtarget_->pt_->dump();   // new line
        rxdb_.print(logtarget_);   // and now dump the data
        return (TCL_OK);
      }

    }
  else {
      fprintf(stdout,"Your logtarget is not defined! You will not be able to\n"
        "print or dump databases (e.g., RxDB). Please create\n"
        "a trace file in your tcl script (Node %u).\n", vanetID_);
    } }
  else if (argc == 3) {
     if (strcmp(argv[1], "lookup") == 0) {
        u_int32_t against = atoi(argv[2]);
        double lh;
        lh = rxdb_.lookup(against);
        return (TCL_OK);
     }
     else if (strcmp(argv[1], "log-target") == 0 ||
              strcmp(argv[1], "tracetarget") == 0) {
        logtarget_ = (Trace*)TclObject::lookup(argv[2]);
        if (logtarget_ == 0) {
           printf("Node %u: logtarget is zero!\n", vanetID_);
           return TCL_ERROR;
        }
       return TCL_OK;
     }
 else if (strcmp(argv[1], "printgroup") == 0) {
                printgroup(atoi(argv[2]));
               return TCL_OK;
                }
  }
  return (Agent::command(argc, argv));
}
void VanetRBCAgent::printgroup(u_int32_t vanetID_) {
                MobileNode *tnode = (MobileNode*)Node::get_node_by_address(vanetID_);
                if(tnode->is_leader_==1)
                {
                printf("\n____________MEMBERS OF %d_____________________\n",vanetID_);
                for(int i=0;i<50;i++){
                        if(tnode->group[i]!=-1)
                        printf("\nGroup member %d is %d\n",i+1,tnode->group[i]);
                }}}
```

```
void VanetRBCAgent::recv(Packet* pkt, Handler*) {
  struct hdr_cmn *hdrcmn = HDR_CMN(pkt);
  struct hdr_ip *hdrip = HDR_IP(pkt);
  struct hdr_vanet *hdrgen = HDR_VANET(pkt);
  if ((u_int32_t)hdrip->daddr() != IP_BROADCAST) {  // check if brdcast mode
    printf("N %u: NOT BROADCAST Packet received!!\n", vanetID_);
    exit(1);
  }
  if(hdrcmn->ptype() == PT_VANETRBC) {
    switch(hdrgen->vn_msgtype) {
      case VANETTYPE_REGBC:
        recvRBC(pkt);
        break;
      default:
        printf("N %u: Invalid VANET packet-type (%d)\n",
                              vanetID_, hdrgen->vn_msgtype);
        exit(1);
    } }
  else {
    printf("N %d: Non-VANET-Packet received (type: %d)\n", vanetID_, hdrcmn->ptype());
    exit(1);
  }
  Packet::free(pkt);
}
void VanetRBCAgent::recvRBC(Packet* pkt) {
  u_int32_t senderID;
  double tStamp;
  double XposS, YposS;   // for storing the Sender's coordinates (example)
  char out1[100];
  struct hdr_vanet_rbc *hdr = HDR_VANET_RBC(pkt);
  senderID = hdr->rbc_senderID;
  tStamp = hdr->rbc_timestamp;
  XposS = hdr->rbc_posx;
  YposS = hdr->rbc_posy;
  rxdb_.add_entry(senderID, Scheduler::instance().clock(), XposS, YposS);
  double trtime = (Scheduler::instance().clock() - tStamp)*1000; // in ms
        MobileNode *msnode = (MobileNode*)Node::get_node_by_address(senderID);
        MobileNode *tmnode = (MobileNode*)Node::get_node_by_address(vanetID_);
        if(msnode->is_leader_ == 1){
        if((tmnode->leader_id_ != msnode->nodeid()))        {
        printf("%d issuing group key to guest-%d time-%f\n",msnode->nodeid(), tmnode- Nodeid(),
tStamp);
        printf("issued group key is %u\n",msnode->group_key_);
          if(tmnode->is_leader_==1) {
                  for(int i=0;i<50;i++){
                                  if(tmnode->group[i]!=-1){
                  MobileNode *mbrnode = (MobileNode*)Node::get_node_by_address(tmnode-group[i]);
                  mbrnode->leader_id_=-1;
                  mbrnode->group_key_=-1;
                  tmnode->group[i]=-1;
                  }} }
 else if(tmnode->leader_id_ != -1) //if the current node is a member of a group  {
                  MobileNode clnode = (MobileNode*)Node::get_node_by_address(tmnode->leader_id_);
                          for(int i=0;i<50;i++){
                                          if(clnode->group[i] == tmnode->nodeid()){
                                                  clnode->group[i]=-1;
                                                  break;
                                          }}}
        for(int i=0;i<50;i++)
        {
                  if(msnode->group[i] == -1)
                  {
                          msnode->group[i]=tmnode->nodeid();
                          break;
```

```
                }
                else
                sprintf(out1, "%s %d",out1,msnode->group[i]);
        }
        tmnode->group_key_ = msnode->group_key_;
        tmnode->leader_id_ = msnode->nodeid();
        tmnode->is_heard_ =1;
        tmnode->is_leader_=-1;
        }
        else
        printf("%d guest heard %d leader time-%f\n",tmnode->nodeid(),msnode->nodeid(),tStamp);
        }}
void VanetRBCAgent::timeout(int) {
   pkt_timer_.resched(interval_);
   sendRBC_pkt();
}
void VanetRBCAgent::sendRBC_pkt() {
   int pktsz;
   Packet* pkt = allocpkt();     // Create a new packet
   struct hdr_vanet_rbc *hdr = HDR_VANET_RBC(pkt);
   hdr_ip* iph = HDR_IP(pkt);
   iph->daddr() = IP_BROADCAST;
   iph->dport() = iph->sport();
   hdr->rbc_msgtype = VANETTYPE_REGBC;  // (necessary for dispatching!)
   hdr->rbc_senderID = vanetID_;
   hdr->rbc_timestamp = Scheduler::instance().clock();
   MobileNode *pnode = (MobileNode*)Node::get_node_by_address(vanetID_);
   pnode->update_position();      // update the position, before using it
   hdr->rbc_posx = pnode->X();    // include current own location
   hdr->rbc_posy = pnode->Y();
   pktsz = hdr->size();           // get packet-size of this type
   hdr_cmn::access(pkt)->size() = pktsz;  // set it in the simulator
```

```
  Scheduler::instance().schedule(target_, pkt,
                           crypto_delay_ + JITTER*jitterfactor_);
  if(running_ == true)
     pkt_timer_.resched(interval_);
}
```

VANETRBC.tcl script :

```
set scriptStart [clock seconds]       ;# start time of the simulation
set maxTripTime -1                    ;# 2 variables for storing the
set maxTripTimeNode -1                ;#     maximum trip-time and node
set fromv -1
set tov -1
set keyv "hello"
set setv -1
set timev 0.0

set val(chan)      Channel/WirelessChannel    ;# channel type
set val(prop)      Propagation/FreeSpace      ;# radio-propagation model
set val(ant)       Antenna/OmniAntenna        ;# antenna type
set val(ll)        LL                         ;# link layer type
set val(ifq)       Queue/DropTail/PriQueue    ;# interface queue type
set val(ifqlen)    50                         ;# max packet in ifq
set val(netif)     Phy/WirelessPhy            ;# network interface type
set val(mac)       Mac/802_11                 ;# MAC type
set val(rp)        DumbAgent                  ;# routing (none)
set opt(tracedir)  out_files/
set opt(outdir)    out_files/
set opt(filename)  $opt(tracedir)out          ;# base filename for traces
set opt(vnfilen)   $opt(filename)-vanet.tr    ;# vanet tracelog
set opt(nn)        10                         ;# number of mobilenodes
set opt(x)         2400
set opt(y)          2400
set opt(sc)         scenario/univ-10.tcl
set opt(stop)       100         ;# simulation end time
set opt(cbrinterv)  5        ;# the interval for sending RBC messages (also
Mac/802_11 set dataRate_            6.0e6
Mac/802_11 set basicRate_           6.0e6
Mac/802_11 set CCATime              0.000004
Mac/802_11 set CWMax_               1023
Mac/802_11 set CWMin_               15
Mac/802_11 set PLCPDataRate_        6.0e6
Mac/802_11 set PLCPHeaderLength_    50
Mac/802_11 set PreambleLength_      16
Mac/802_11 set SIFS_                0.000016
Mac/802_11 set SlotTime_            0.000009
Phy/WirelessPhy set RXThresh_   6.72923e-11     ;# 300m at 5.15e9 GHz
Phy/WirelessPhy set freq_       5.15e9
Phy/WirelessPhy set Pt_         0.281838        ;# value for the 300m case..
proc finish {} {
   global ns_ tracefd
   global ns_ vanettracefd
   global ns_ namfd
   global opt
   global scriptStart
   global maxTripTime
   global maxTripTimeNode
   $ns_ flush-trace
   close $tracefd
   close $vanettracefd
   close $namfd
   exec nam $opt(filename).nam
   exit 0
 }
```

```
Agent/VanetRBC instproc recv {key} {
  global setv fromv tov keyv timev
  set keyv $key
  set setv 1
}
set ns_ [new Simulator]                    ;# simulator object
set tracefd [open $opt(filename).tr w]         ;# set up trace file
$ns_ trace-all $tracefd
set namfd [open $opt(filename).nam w]          ;# set up nam trace file
$ns_ namtrace-all-wireless $namfd $opt(x) $opt(y)
set vanettracefd [open $opt(vnfilen) w]
set VanetTrace [new Trace/Generic]
$VanetTrace attach $vanettracefd
set topo   [new Topography]                    ;# create the topology
$topo load_flatgrid $opt(x) $opt(y)
create-god  $opt(nn)
$ns_ node-config -adhocRouting $val(rp) \
   -llType $val(ll) \
   -macType $val(mac) \
   -ifqType $val(ifq) \
   -ifqLen $val(ifqlen) \
   -antType $val(ant) \
   -propType $val(prop) \
   -phyType $val(netif) \
   -channel [new $val(chan)] \
   -topoInstance $topo \
   -agentTrace ON \
   -routerTrace OFF \
   -macTrace OFF \
   -movementTrace OFF
set temp {blue red green gold brown yellow}
for {set i 0} {$i < $opt(nn)} {incr i} {
   set node($i) [ $ns_ node $i ]
        $node($i) color [lindex $temp[expr $i]]
}
for {set i 0} {$i < $opt(nn)} {incr i} {
   set p($i) [new Agent/VanetRBC $i]
   $ns_ attach-agent $node($i) $p($i)
   $p($i) log-target $VanetTrace
}
for {set i 0} {$i < $opt(nn)} {incr i} {
   $p($i) set interval_ $opt(cbrinterv)
   $p($i) set jitterFactor_ 0.001
   # for example (ECC): sig-delay (0.003255) + 2*verif-delay (0.00762)
   $p($i) set crypto_delay_ 0.018495
}
source $opt(sc)
for {set i 0} {$i < $opt(nn)} {incr i} {
    set sttime [expr {5 + $opt(cbrinterv)*$i/$opt(nn)}]
    $ns_ at $sttime "$p($i) start-regbc"
         $ns_ at $sttime "$node($i) set count_ 0"
         $ns_ at $sttime "$node($i) set is_leader_ 0"
         $ns_ at $sttime "$node($i) set group_key_ 0"
         $ns_ at $sttime "$node($i) set leader_id_ -1"
 }
 proc display {} {
 global ns_ setv fromv tov keyv timev
 for {set i 0} {$i < 10} {incr i} {
    global $p($i)
 }
 set now [$ns_ now]
 set result [expr {($setv==1)}]
 puts "setv=$setv timev=$timev result=$result"
 if {$result==1} {
```

```
$ns_ at $now "$ns_ trace-annotate \"$keyv\""}
$ns_ at [expr $now+2] "display"}
set num $opt(nn)
set key 0
set k 0
proc check {} {
global ns_ node key k p num
set now [$ns_ now]
for {set i 0} {$i < $num} {incr i} {
        set hd [$node($i) set group_key_]
        set cnt [$node($i) set count_]
        if {$hd == 0} {
                if {$cnt == 1} {
                $ns_ at $now "$p($i) assignkey"
                $ns_ at $now "$node($i) set count_ 0"
                break
                } else {
                set cnt [expr $cnt + 1]
                $ns_ at $now "$node($i) set count_ $cnt"
                }}}
        $ns_ at [expr $now+5] "check"}
$ns_ at 20.0 "check"
for {set i 0} {$i < $opt(nn)} {incr i} {
        $ns_ at 100 "$p($i) printgroup $i"
  $ns_ at 100 "$p($i) stop-regbc"
}
$ns_ at $opt(stop) "finish"
$ns_ run
```

APPENDIX 3 - LIST OF STANDARD NOTATIONS USED :

Notation	Description
i	A entity/node in the VANET.
$i \rightarrow j$	Entity i broadcasts to entity j.
G_j	A group j of nodes in the VANET.
$\mathcal{N}$	Set of all n nodes in the VANET, i.e. $\lvert\mathcal{N}\rvert = N$.
$\mathcal{G}$	Set of all g groups in the VANET, i.e. $\lvert\mathcal{G}\rvert = g$.
$\mathcal{H}$	Set of groups in the VANET. $\mathcal{H} \subseteq \mathcal{G}$.
L_{max}	Maximum size for a group.
GL_j	Group Leader of group G_j.
GID_j	Group ID of group G_j.
$PID_{i,k}$	k^{th} pseudonym of node i. Each node i has a set of w pseudonyms, $\{PID_{i,k}\}_{k=1}^{w} = \{PID_i\}$.
A_{GL_j}	ID of GL_j. Note that $A_{GL_j} = GID_j \Vert 0^y$, where y is size (in bits) of node ID field.
A_{aa_j}	LBS application access address selected from an address range for group G_j.
$A_{i,j}$	ID of node i that is a member of group G_j. Note that $A_{i,j} = PID_{i,k}$ or $A_{i,j} = GID_j \Vert A_{aa_j}$.
$A_{broadcast}$	Broadcast address for network.
$A_d \Vert A_s \Vert data$	Destination address $\Vert$ Source Address $\Vert$ Data.
$speriod$	Random silent period. $speriod_{min} \leq speriod \leq speriod_{max}$.
s_{min}, s_{max}	Minimum and maximum speed limits for a node.
R_{max}	Maximum number of broadcast repetitions.
T_{max}	Maximum waiting period for an ACK or a reply.
W_{max}	Maximum waiting period for a group join request.
$x \Vert y$ or (x, y)	x concatenated to y.
$\{x\}$	A set of elements.
/** comment **/	Comments in the pseudocode.
K_x, K_x^{-1}	Public and private key pair of entity x.
$k_{x,y}$	Pairwise symmetric key of two entities x, y.
k_{G_j}	Symmetric key of group G_j.
$c = E_{K_x}(m)$	Encryption of message m with public key K_x.
$D_{K_x}(c)$	Decryption of ciphertext c with private key K_x^{-1}.
$E_{k_x}\{\cdot\}$, $D_{k_x}\{\cdot\}$	Encryption and Decryption with symmetric key k_x.
$sign_i(m)$	Digital signature on message m with private key of entity i.
$h(m)$	Cryptographic hash of a message m. Also, $h^n(m) = h(h^{n-1}(m))$, $n \geq 2$.
q_i	A secret quantity of node i.

Table 7.1 List of Standarad Ntations Used

REFERENCES

8. REFERENCES:

1. Beresford.A.R, "Location privacy in ubiquitous computing," Ph.D. dissertation, University of Cambridge, November 2004
2. Camenisch.J and Lysyanskaya.A, "An efficient system for nontransferable anonymous credentials with optional anonymity revocation," in *Advances in Cryptology - EUROCRYPT 2001*, ser. LNCS, vol. 2045. Springer, 2001, pp. 93–118.
3. Chaum.D, "The dining cryptographers problem: Unconditional sender and recipient untraceability," *Journal of Cryptology*, vol. 1, pp. 65–75, 1988.
4. Daniel Jungels, Laboratory for computer Communications and Applications (LCA) EPFL, VANET-Skeleton for ns2. December 23, 2005
5. Doetzer, F., Kosch, Strassberger.M. , Classification for traffic related intervehicle messaging. in Proceedings of the 5th IEEE International Conference on ITS Telecommunications, Brest, France (2005)
6. Dotzer.F, Kohlmayer.F, Kosch.T, and Strassberger.M, "Secure communication for intersection assistance," in *Proc. of the International Workshop on Intelligent Transportation (WIT)*, 2005
7. Fitzmann, A., Kohntopp, M., Anonymity, unobservability, and pseudonymity - a proposal for terminology. In Federrath, H., ed.: Proceedings of Workshop on Design Issues in Anonymity and Unobservability, Springer (2001)
8. Florian Dotzer , Privacy Issues in Vehicular Ad Hoc Networks BMW Group Research and Technology, Hanauerstrasse 46, 80992 Munich, Germany , florian.doetzer@bmw.de
9. Florian Dotzer, "Privacy issues in vehicular ad hoc networks," in *Proc. of the Workshop on Privacy Enhancing Technologies (PET)*, 2005.
10. Francisco J. Ros Pedro M. Ruiz *{fjrm, pedrom}* @dif.um.es *Dept. of Information and Communications Engineering University of Murcia* Implementing a New Manet Unicast Routing Protocol in NS2 December, 2004

11. Gruteser.M and Grunwald.D, "Anonymous usage of location-based services through spatial and temporal cloaking," in *Proc. of the ACM MobiSys*, 2003, pp. 31–42.

12. Huang.L, Matsuura.K, Yamane.H, and Sezaki.K, "Towards modeling wireless location privacy," in *Proc. of the Workshop on Privacy Enhancing Technologies (PET)*, 2005.

13. Hubaux.J.P, Capkun.S, Luo.J, Security and privacy of smart vehicles. IEEE Security & Privacy 2 (2004)

14. Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta G. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proceedings of MobiCom'98*, pages 85–97, Oct. 1998.

15. Markus Jakobsson. Privacy vs. Authenticity. PhD thesis, University of California at San Diego, 1997.

16. Magda El Zarki, Sharad Mehrotra, Gene Tsudik, and Nalini Venkatasubramanian. Security issues in a future vehicular network. In European Wireless, 2002.

17. NS Manual (formerly *ns* Notes and Documentation).Available at http://www.isi.edu/nsnam/ns/doc/.

18. Raya.M and Hubaux.J.P, "Security aspects of inter-vehicle communications," in *Swiss Transport Research Conference*, 2005.

19. Sastry Duri, Marco Gruteser, Xuan Liu, Paul Moskowitz, Ronald Perez, Moninder Singh, and Jung-Mu Tang. Framework for security and privacy in automotive telematics. In Proceedings of the 2nd international workshop on Mobile commerce, pages 25–32. ACM Press, 2002.

20. Zarki.M.E, Mehrotra.S, Tsudik.G, and Venkatasubramanian.N, "Security issues in a future vehicular network," in *European Wireless*, 2002.