

Abstract

Vehicle-to-vehicle communication (V2V) is the process of transmitting data wirelessly between vehicles. By relaying the information about location and speed data between vehicles is via an ad hoc network is the primary aim of V2V communication. This communication takes place over an ADHOC mesh. V2V communication is thought to be the better alternative for the current vehicular(original equipment manufacturer: OEM) embedded systems which have features like adaptive cruise control, rear parking sonar and backup camera because the V2V technology enables an ubiquitous 360-degree awareness of surrounding threats. With the further advancement of the v2v communication many security flaws were found. The best way to analyze these flaws would be through a bi-directionally coupled simulation environment made of OMNET/SUMO, ns2 /SUMO, etc. Many possible attacks like Botnet, GHOST, congestion etc. are being used to exploit the system every day, which can be simulated and studied in the simulated environment. In the starting we discuss about the general VANET characteristics, in the second part we discuss about all the existing flaws with which the system can be attacked and in the third and last part we discuss about the solutions to those problems and how the present conditions can be improved through the implementation.

Chapter 1

INTRODUCTION

Autonomous vehicles platooning has been a key subject of importance in the 21st century because of its ability to benefit road transportation, improving traffic efficiency, enhancing road safety and reducing the fuel consumption of the vehicles. A Special class of ad-hoc routing based network known as VANET i.e. vehicular ad hoc network has been of keen significance in road transportation and security applications. Registration and management of the individual identities of the driver are handled by the On-Board units (OBUs) and Roadside units (RSUs). The RSUs have been installed to gather information and the vehicles that are handled by the VANET are primarily installed with the OBU's. Such vehicles are able to traverse freely on a specified modular road network, and are able to communicate with each other or with RSU's and other identified authorities. The communication channel could be either a single or multi-hop mode of data transmission using the DSRC (Dedicated Short Range Communication) on V2V or V2X. V2V communication is technically a form a large-scale distributed embedded systems. In recent future most of the vehicles are subjected to be equipped with wireless on board units GPS (Global Positioning System), EDR (Event Data Recorder), (OBU), and sensors (radio-detection units RADAR and LIDAR/LADAR) as shown in Figure 1.1. Traffic congestion status is measured using such equipments and other statistical data is also recorded and then automatically necessary actions are taken in the vehicle itself and the information is relayed to other vehicles and RSU's.

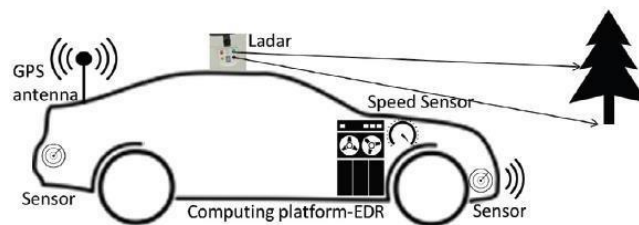


Fig. 1.1 Car components

The system which exists in the present day works more like a WAN or Internet where each car on the network acts like a computer. In today's world almost all cars are equipped with a form of digital circuit inside it with CPUs to take certain important decisions (OBU) for e.g. when the car

speeds up your seat belts get tighter or when the car detects that it's raining it automatically starts the wiper etc. The car communicates to each other like how computers communicate to each other on a network and find a solution to make the whole journey more efficient and safe.

ITS intelligent transportation system:

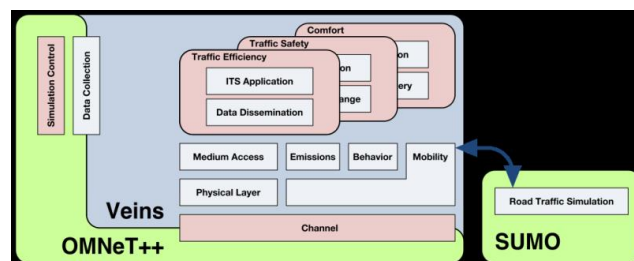
Autonomous vehicles platooning has received considerable attention in recent years, due to its potential to significantly benefit

- Road transportation
- Improving traffic efficiency
- Enhancing road safety and
- Reducing fuel consumption

The Vehicular impromptu Networks and the true vehicular systems administration standard IEEE 802.11p,4G,5G openness convention are absolutely vital apparatuses for the organization of platooning applications, since the participation among vehicles depends on a solid correspondence structure. Be that as it may, vehicular systems can endure distinctive security dangers. In fact, ITS are somewhat expansive scale circulated installed frameworks which are huge scale in nature and perpetually geologically dispersed. Diverse kinds of versatile computational gadgets are then prepared on the vehicles or physical framework to influence them to see neighborhood condition states and occasions. Furthermore, these gadgets are associated and teamed up with one another to play out specific assignments. ITS on location condition is supplanted by a reproduction situation.

ITS is actualized utilizing Simulation systems

Using simulators as **OMNET++** and **SUMO** and **NS2** (network simulator 2) which is **the bidirectional coupled environment** for running VEINS (open source framework for running vehicular network simulations)OMent++-veins and SUMO work in the following manner to implement the simulations. Fig 1.2 omnet-veins-architecture



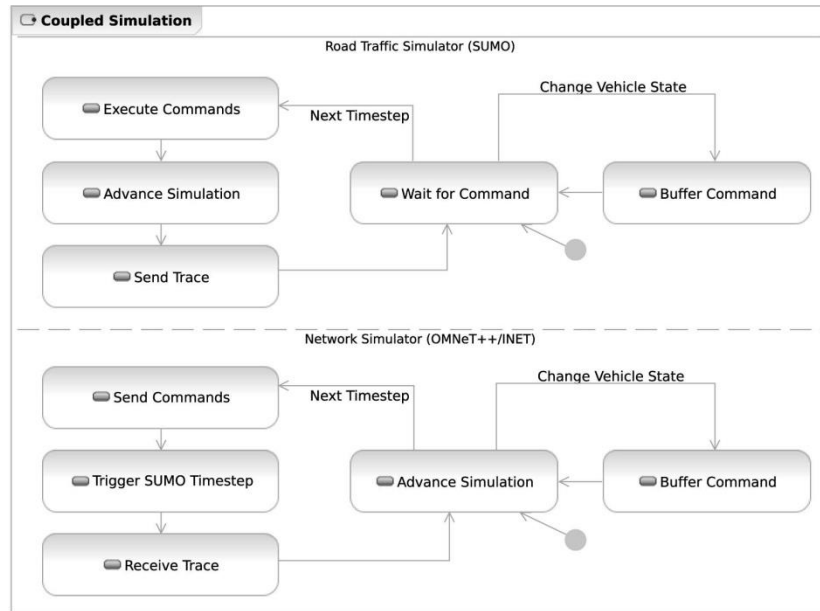


Fig 1.3 Simulation events

NS2 simulator is being used to simulate the network scenario which we create and obtain the results and depending on the work mechanisms of the attack and the simulation results, defense mechanisms against the attacks are being developed.

Ns2 and sumo are used to produce the simulation whose results are generated and stored in trace file called **.tr** (trace file).

The sumo helps in converting the .xml file of the map into .net.xml file which could be implemented with the various networking scripts and tools to produce the important .tcl file which is compiled and run by the NS simulator.

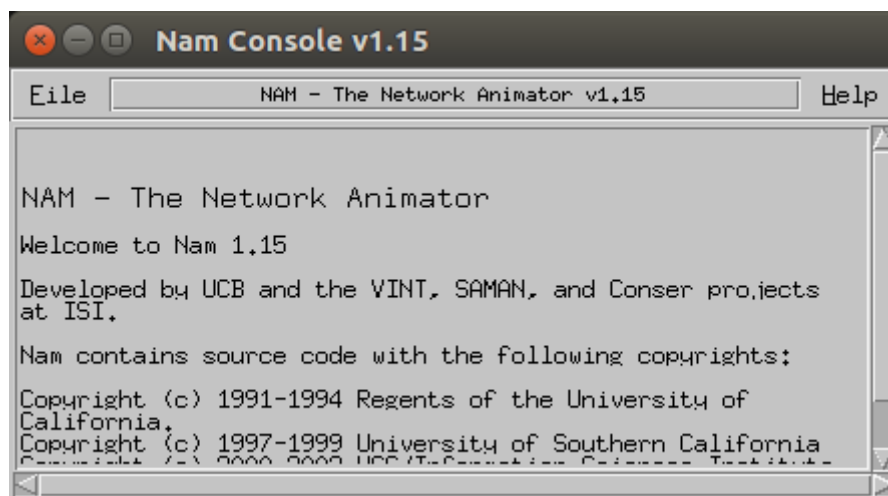


Fig 1.4 NS-2 software

Chapter 2

LITERATURE SURVEY

1. Cognition attacks to autonomous cars using vehicular botnets

The main purpose of this paper is to not simply induce the congestion attacks but rather use it as an example to demonstrate the power of vehicular botnets thus shows the possibility of extending research in the field of security development in vehicular simulation.

2. Ghost: Conceding vehicular bot net communication in the vanet control channel

Structure and exhibit vanet based cap correspondence convention that conceals itself in progressing system traffic over the control channel, we demonstrate the above through system reenactment demonstrates the likelihood of broadening research in the field of security advancement in vehicular reproduction

3. A sumo based evaluation of road accidents impact on traffic congestion level in smart cities

Evaluation of road accidents and its impact on traffic congestion level via sumo. A graph comparison of vehicular average increase in time of rerouting accidents to topology average, traffic density, accident duration and no of accidents performance which shows the possibility of extending research in the field of security development in vehicular simulation

4. Security challenges in intelligent transportation system

To find the existing security flaws in DDS system configuration parameters present in the base ITS protocol system. A graphical measurement for finding flaws in the network security communication protocol. Shows the possibility of extending research in the field of security development in vehicular simulation

5. Realistic traffic urban simulation as VANET via veins framework

To simulate custom locations and custom parameters via omnet and sumo traffic interface. Shows the possibility of extending research in the field of security development in vehicular simulation. Network simulation parameter objects Omnet++, Sumo++, Sumo configuration files.

6. Bidirectional coupled and road traffic simulation for improved IVC analysis

As the decision of a flexibility model effects the consequence of propagations, everything considered, the use of an operator model is basic for making significant appraisal results. In perspective on these recognitions, we developed the blend reenactment framework Veins (Vehicles in Network Simulation), made out of the framework test framework Omnet++ and the road traffic test framework SUMO. In a proof-of-thought consider, we show its positive conditions and the necessity for bidirectional coupled reenactment subject to the evaluation of two shows for event alerted over VANETs. With our made framework, we can push the front line in execution evaluation of IVC and offer expects to survey made shows even more exactly.

7. Prediction of vehicle collision probability at intersection

To exhibit the circumstances and logical results of impact and mishaps and to benefit security parameters. Vehicle crash location and blockage control are prime difficulties to be met. Numerous advancements are in real life for crash free traffic.

8. Management Security in intelligent transport system

Innovative security designing for managing security in Intelligent Transport Systems (ITS), in light of our examination finished as a noteworthy part of the so far nonstop ICSI. Exhibits the probability of expand ing research in the field of security improvement in vehicular entertainment.

9. Exploiting daily trajectories for efficient routing in vehicular ad hoc routing networks

In this paper, they shed light on the prerequisite for instruments that consider the vehicles trajectories and it displays a depiction that shows the spatiotemporal typicality of the vehicle advancement. The makers propose another framework for perceiving the spatiotemporal association dispatch between vehicle headings and to make a novel method named ROSTER for unicast guiding in pitiful VANETs. Proliferations results exhibit that the proposed game plan astonishingly lessens message overhead in the framework by keeping up great elements of transport rate in connection with various shows...

Chapter 3

EXISTING SYSTEM

The system which exists in the present day works more like a WAN or Internet where each car on the network acts like a computer. In today's world almost all cars are equipped with a form of digital circuit inside it with CPUs to take certain important decisions for e.g. when the car speeds up your seat belts get tighter or when the car detects that it's raining it automatically starts the wiper etc. The car communicates to each other like how computers communicate to each other on a network and find a solution to make the whole journey more efficient and safe. These cars are equipped with some form of radio transmission devices or 4G/5G LTE communication devices which help it with the communication part. The very basic forms of communications can be seen in the diagram where we have v2v, v2x ($x = i, v$) and i2v type of communications.

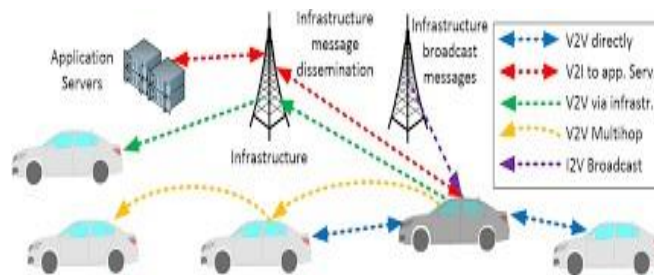


Fig. 2.1 All types of communication

VANET attributes related to Network-Topology and Communication-Mode

- 1) Large Scalable networks: As discussed VANET is a form of WAN and can be implemented for large cities/multiple cities or even a country as a whole. Thus requires regular and immense management of security protocols.
- 2) Wireless communication: The vehicles (nodes in a WAN) are connected through wireless channel and exchange large amount of data thus requires highly-secured network.
- 3) High mobility and rapidly changing network topology: Nodes are able to move at a very high

speed and more so they move randomly thus it becomes burdensome to predict their ideal positions and the topology of the specific network. It leads to *disconnection issues and impossibility of handshake*.

VANET Characteristics related to Drivers and Vehicles

- 4) *High CPU power and extra processing energy*: The vehicles are equipped with powerful OBUs which have powerful CPUs and memory just like a computer and have batteries on which the system runs the highly complex cryptographic calculations.
- 5) *Calculating time and position*: GPS is a common installment on numerous vehicles as many applications rely on position and geographical addressing. A non-tamper able system ensures that the location of nodes is kept private from the attacker.
- 6) *The nature of the participants*: Larger proportions of drivers are considered to be good and law abiding citizens and are regular users of the service.
- 7) *Prevailing law enforcement agencies*: The malicious attacker can be caught by the law enforcement services.
- 8) *Central enrollment with standard support and investigation*: vehicles have a one of a kind id (tag). Vehicles experience intermittent upkeep check for programming/equipment and firmware refreshes. In PKC (Public key Cryptography), support is for refreshing keys, Certificates and refreshing CRL with crisp CRL (Certificate Revocation List).

VANET Security Challenges and some Limitations

On the very basic level it is very similar to a computer network where u have computer/cars communicating with each other and like all computer networks this system can be exploited in various forms. In VANET, security must guarantee that the exchanged messages are not maliciously modified by the attackers, also the driver's system should notify the traffic environment-parameters correctly within the given time period.

Security Challenges

- 1) *The Network size*: The large geographical area, the high mobility, the dynamic nature of the topology, the short connection duration and the frequent disconnections is a major security issue .The size of the network is large and grows uncontrollably but it is scalable in nature

and continues to grow without any global authority to govern its standards.

- 2) *The information verification*: Trust is one of the core requirements of the system as there is continuous exchange of data between nodes and central systems, there should be a proper system to verify the authenticity and integrity of the data. *Trustworthiness of the data* is more useful compared to that of the nodes transmitting it.
- 3) *Distribution of keys*: Security mechanisms use cryptographic keys, which make their secure-distribution highly important.
- 4) *The Forwarding algorithms*: used to transfer the data in the best manner to all the nodes in the system, whether you are using a UDP broadcast communication (v2v based on configured hops) or communication with a central authority to get the required road conditions, weather factors etc.

Limitations

- 1) *Probability involved in algorithms*: Every algorithm's probability has some error which can affect the outcome and people's life.
- 2) *Environmental influence*: due to the magnetic waves, the channel in which the data exchange is going on gets affected and it would lead to loss of data, handshake errors. The above mentioned limitations can be better handled in a proper manner if we implement the required properties properly.

Threats to the Wireless Interface of the driver

- 1) Identity and geographical position tracking (Location-time Tracking): The driver is breached by the attacker in order to track and gain important authentication information from the individual who could be used for malicious activity. For example, some of the rental car companies use their vehicular tracking system to keep records of their customer's movement for various malicious activities, which is a major breach of the privacy of the user.
- 2) DoS (Denial of service): Resources and the various services of the v2v are made unavailable to the user network by a malicious attacker. This can take place by either blocking the physical channel or by "Sleep Deprivation DoS".
- 3) DDoS (Distributed Denial of Service): DOS attacks occurring from various geographical

locations to the network are considered to be a form of DDOS attacks. It can make the channel unavailable or drain out the power through sleep deprivation based DDoS attacks.

- 4) Sybil Attack: In this attack a similar identity is issued to multiple vehicles travelling on road hence an illusion is presented to other vehicles when a wrong message is sent to the vehicles ultimately benefitting the attacker.
- 5) Malware Infection: a node/attacker inside the network (in a car) transmits spam messages into the network to increase latency and bandwidth consumption of the network. Due to the lack of necessary security-controls in the infrastructure and centralized administration it becomes cumbersome to control such kind of attacks. Spam messages may be disseminated by the attacker to a group of multiple users, and these messages are rendered useless by the users similar to advertisement messages, it may contain certain malwares in it which could trigger malfunction behavior in the system. For e.g. Broadcasting RAIN weather messages over the network to different cars would trigger the windshield unnecessarily and cause driving nuisance to the driver. It may sometimes contain malicious data in it which could infect the OBUs of the vehicles with viruses and Trojans.
- 6) Man in the Middle Attack (MiM): While communication is established between two vehicles a malicious node is able to listen to the data exchange and after gaining certain authentication information required for communication it communicates with each of them and presents false information to each other and causing various nuisance to the driver by sending unnecessary messages.
- 7) Brute force Attack: the attacker tries to get user-personal information such as password or PIN or to decrypt the data, or to validate network security by the help of trial and error based algorithms which directly attacks the authentication part of the networking system. It includes algorithms like rainbow-table based attacks or basic dictionary attacks to crack into the system.
- 8) Black Hole Attack: Shortest path algorithms are used by the network to send data, AODV protocol is such a protocol. In this attack the attacker after getting into the network pertains to be the part of the shortest route to each vehicle by broadcasting wrong location information due to which the vehicles get tricked into transmitting the messages to them which the attacker after obtaining them obtains the important information in them and later on drops

the packet, leading to the failure of information exchange between particular nodes.

Threats related to Hardware and Software

Other than the threats presented like Sybil attack, DoS, Malware, spam, Brute force and Man in middle mentioned above in sub-section (1), we can list

- 1) Message Suppression or alteration: By understanding the software and hardware used in the OBUs and RSUs, one can trigger certain malfunction in them. By exploiting certain messages either by suppressing them or altering them or even delaying them could trigger unnecessary response from the software used in the RSUs and OBUs. The hardware can also be exploited by altering various messages. Thus to avoid this the only way is to authenticate the message being transferred and coming up with some mechanism which would make the message opening/capturing difficult for attackers
- 2) Assuming the identity of a node (Spoofing or Impersonation or Masquerade): The attacker impersonates as a specific node after getting into the network and this procedure is known as the black hole problem. To gain access to restricted messages and to avail privileges of the user profile the attacker declares itself to be a good/normal node as the culprit gains unauthorized access, this is a type of replay attack where attacker tries to imitate a legitimate user/RSU by using earlier generated/used frames from the communication channel in the new connections which it establishes with various nodes/vehicles. There are various forms of spoofing which exploits the software vulnerabilities to gain certain kind of privileges in the network. By understanding the hardware and software vulnerabilities one can exploit it by spoofing in the correct manner.
- 3) Tampering with the Hardware of the vehicle: This is one of the major breaches as the employees fiddle with the hardware in a malicious during the yearly maintenance of the vehicle, in the vehicle manufacturers service centers, it is generally done to either to get or put special data into the OBUs of the car , for *e.g.* A Trojan can be installed on the main CPU board of the car to compromise the control of the driver on the car, one can trigger brake clutch, almost all the things through the circuit of the car. We have to understand a car as a form of computer on

the network which can be infected in the similar manner with viruses, which compromises the user's control over his/her system, allowing the attacker to control the car. Some worst cases could be that he could power-off the car on a high speed national highway. Not only manufacturer's employee but a lot of hackers know how to install such viruses onto the circuit board of a car.

- 4) Wormhole attack: Overhearing data; it is almost a form of black hole attack where the malicious node (insider node generally) impersonates to be the closest node and tunnels the message to some other part of the network causing various software malfunctions on the car like drop of important authentication messages etc. As you can see in the figure X' node takes in the data by acting as the nearest node and then redirects it to some other point. (form of routing attacks)

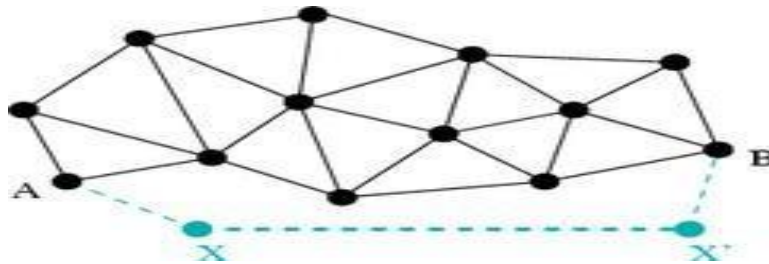


Fig. 3.2: wormhole attack

- 1) Greyhole Attack: A misleading malicious node programmed by the attacker causes the network to transfer the data packet forward to other hosts but occasionally this affected node retires for a while and conforms to its normal behavior, Basically it performs a blackhole attack or a wormhole attack but switches its behavior from time to time to confuse the software's security protocol.
- 2) Cheating with position and timestamp information (GPS Spoofing): false location and time lapse is generated by vehicles causing various kinds of accidents by giving rise to falsified GPS locations one can cause wrong messages to be delivered to the driver.
- 3) Timing Attack: A delay is inserted before forwarding the received messages by addition of invalid timeslots created by the malicious vehicles receive the messages after a delayed period or after a particular event has occurred for which the information was generated.

Threats to Sensor of Vehicles

Other than GPS spoofing as mentioned earlier in section (2) some other attacks are:

- 1) Illusion Attack: The sensors reading of the vehicle are purposely deceived by the attacker in order to perceive wrongful data to the neighbors are subjected to failure and hence leads to attainment of wrongful information by the user there are various machines that can be used to make and broadcast the sensor readings.
- 2) Jamming Attacks: the radio frequencies used by the vanet nodes for sensory purpose are being jammed, sensors for moisture readings are jammed, and the sensors behind the parking are sometimes fiddled with. Hence may lead to loss of data.

FAMOUS SECURITY INFRASTRUCTURES BEING USED.

The underlying foundation that is responsible for the framework of the system is known as an *infrastructure*.

It is a key component in development of the Security architecture hence provide an efficient security design. It delineates the utmost potential risks that may be specific to certain environment and enables application of security control when required.

Here we describe PKI i.e. public key infrastructure because of its immense popularity.

Public key Infrastructure

Public key infrastructure is one of the most common form of security enabled on vast WAN type of networks, it helps in the identification and distribution of public encryption keys. Data is exchanged in a secure fashion over the routing networks which helps to verify the personal identities of the other party. PKI comprises of software, hardware, various policies and standards. These in unison are responsible for the revocation, administration, distribution and creation of keys and digital certificates used in the security infrastructure. The components included in the PKI are as follows:

A trustworthy root parameter, called a Root certificate authority (CA), is considered to be a potential trusted parameter and is capable of providing services to authenticate the identities of the entities.

A subordinate CA which holds the duty of registrations and is, certified by a root CA. The root permits the use of certificates for specific purposes. Its primary purpose is the protection of the root CA. Attacks can be checked before reaching the root CA as are first supposed to pass

through the subordinate CA. It can be accessed by both root and subordinate CA.

Issued certificated and private keys are present on each vehicle in the form of a 'certificate store'. This is what the basic idea for the PKI presents, the high level of security architecture is used by adding more layers of authentication In Europe and USA, and they have built their own VANET-security architecture using PKI. In Europe(EU) there exists a different PKI architecture as shown by ETSI in [12] which defines the important security aspects for intelligent transportation system .In USA, the VSC-A (Vehicle Safety Communications- Applications) authority, has considered the usage of NHTSA (National Highway Traffic Safety Administration) [11] with its security architecture for VANET.

Chapter 4

ALGORITHMS

Algorithms/protocols used by (a) veins: (specifically for veins-omnet-sumo):

Two different protocol variants were used for evaluating VANETs in the Veins environment.

These two variants represent extreme settings for possible Inter Vehicular Communication solutions.

1. Centralized TIC Scenario Relying on TCP and Standard MANET Protocols
Conventional TICs are organized in a centralistic way. Usually, sensor based traffic monitoring systems are deployed directly at the roadside to collect information about current traffic conditions.
 - Moreover, vehicles may take part as sensors in this situation. The checking information is exchanged to a focal TIC, where the present street circumstance is investigations. The consequence of this circumstance examination is transmitted to every single partaking vehicle utilizing a communicate medium.
 - Normal for a concentrated TIS is that the traffic data is prepared in at least one devoted (unified) traffic handling elements, e.g., a TIC.
 - Figure shows the centralized form where the RSU communicates with the vehicles and guides the accordingly by warning them of road traffic ahead.

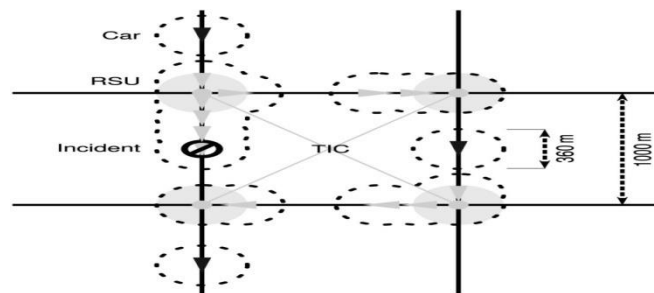


Fig 4.1 Centralized TIC scenario

1. Distributed/Self-Organized TIS Using UDP Broadcast Communication

- An option and totally unique methodology for observing the traffic circumstance and circulating the traffic messages to vehicle drivers is to utilize a decentralized self-sorting out TIS.
- A model is the decentralized SOTIS approach, in which vehicles illuminate each other of the nearby traffic circumstance by methods for IVC.
- The traffic circumstance examination is performed locally in every vehicle. No correspondence/sensor framework is required. For a worldwide course advancement, the SOTIS

- Has a destination sequence number for preventing of routing loops and avoidance of old and broken routes.

2. Source broadcast RREQ data packet for searching route

<source_addr, source_sequence#, broadcast_id, dest_addr, dest_sequence#, hop_cnt>

3. Destination replies using RREP (Route Reply) unicasting

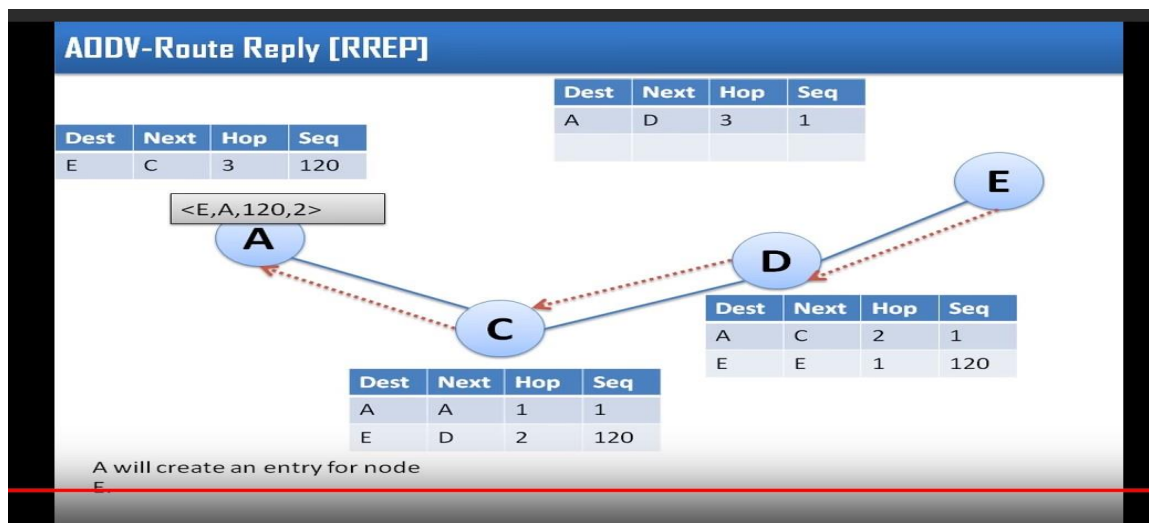
<source_addr, dest_addr, dest_sequence#, hop_cnt, lifetime>

RREP contains the current sequence number, hop count = 0, full lifetime

4. Intermediate Node

Discard duplicate packet

Send RREP if it has active route with higher sequence number.



Chapter 5

IMPLEMENTATION

- Set up NS2 NAM by downloading it from source forge as the libraries are corrupted download the various dependencies required to download NS2
- Download NS2 by the local file name on internet by making use of source forge libraries, NS-all in one -2.3.5 file on the website, follow the procedure and download the software.
- Download various dependencies of NAM and ns simulator.
- Download OSM map from maps.org that converts real time maps into xml file. Those xml files are stored in the .OSM format and location can be searched easily such as the 'Kati Para' bridge in test.
- Now manually select the locations in the map i.e. roadways, highways.
- Create a home directory /test in home itself, in that directory save the OSM map download from map.org as test.osm.

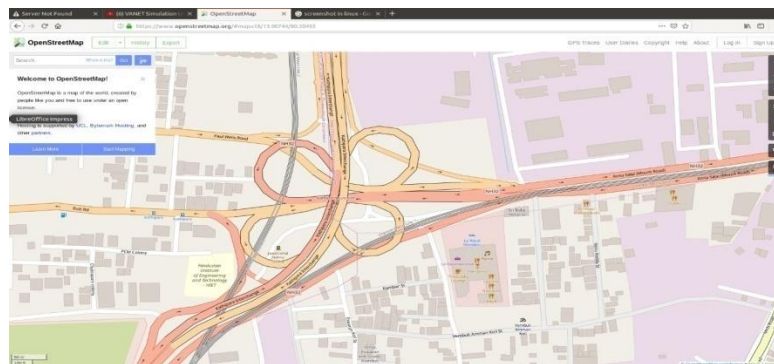
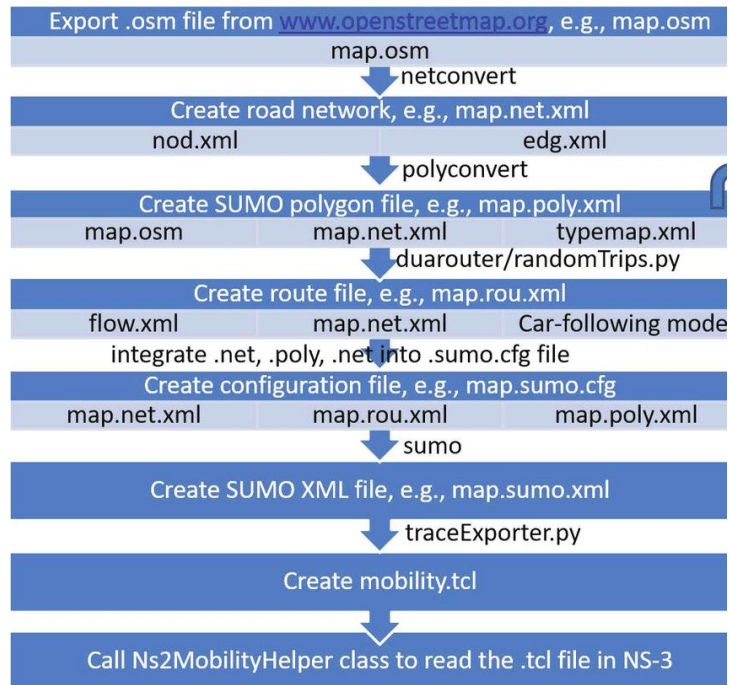


Fig 5.1 www.openstreetmap.org

- The use of
- randomTrips.py
- traceExporter.py
- Simple-wireless.tcl file

All the above are used from sumo/tools and sumo/data.



- Now install sumo
- Earlier stable version was used but now a unstable version is available on the public:ppa/sumo:stable 1.1.0 ver , simulator which causes the system to crash.
- In the current build edition sumo build 1.1.0 ,nam 1.15 ns 2.35
- After successful installation from the sudo apt- get ppa sumo/stable.
- In the current model we will execute the current map using sumo and AODV protocol is used where each node acts as cars moving in the traffic.
- It is started by saving the map as test.osm in the /test folder then and the using the the netconvert tool as :

netconvert --osm-files test.osm -o test.net.xml

aryeshk@aaryeshk-GL503VM:~/test\$ cd data/typemap/

```

aaryeshk@aaryeshk-GL503VM: /usr/share/sumo
Warning: Ignoring restriction relation '2741319' with unknown to-way.
Warning: The traffic light '256127378' does not control any links; it will not be build.
Warning: Could not build program '0' for traffic light '256127378'
Warning: Speed of straight connection '84084670#0_0->84084670#1_0' reduced by 10.05 due to turning radius of 26.92 (length=11.25 angle=39.67)
Warning: Speed of straight connection '28077971_0->214413823_0' reduced by 6.53 due to turning radius of 82.10 (length=22.42 angle=30.80)
Warning: Speed of straight connection '28077971_1->214413823_1' reduced by 7.30 due to turning radius of 76.25 (length=20.81 angle=30.80)
Warning: Speed of straight connection '23641748_1->28078045#1_1' reduced by 5.41 due to turning radius of 90.97 (length=15.26 angle=24.69)
Warning: Intersecting left turns at junction '256119290' from lane '23641255_0' and lane '204209805#1_1'. (increase junction radius to avoid this)
Warning: Speed of straight connection '27769229#1_0->28071747#1_0' reduced by 12.36 due to turning radius of 43.24 (length=18.16 angle=39.52)
Warning: Speed of straight connection '27769229#1_1->28071747#1_1' reduced by 12.86 due to turning radius of 40.48 (length=16.98 angle=39.52)
Warning: Speed of straight connection '25646322#1_0->-83826841#2_0' reduced by 10.73 due to turning radius of 18.55 (length=18.56 angle=74.88)
Success.
aaryeshk@aaryeshk-GL503VM: ~/test$ cd
aaryeshk@aaryeshk-GL503VM: ~$ cd $SUMO_HOME
aaryeshk@aaryeshk-GL503VM: /usr/share/sumo$

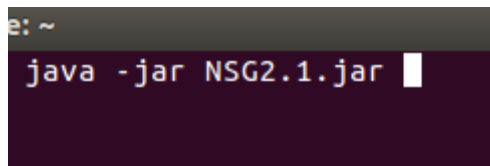
```

- Now create the test.sumo.cfg file which can be used to create the sumo-simulation which will in turn help us to define the routes for ns2 simulation.
- Create test.sumo.cfg file with the above code printing towards the .net.xml file and rou.xml file

size and scale of custom maps we are going to use a simple tcl file for simulating a blackhole attack and in a similar manner you can simulate other custom attacks depending on the methodology of the attack. Identification of TCL packets in a routing network of nodal points with the help of AODV protocol. A packet was dropped in the generic space and its TCL output was traced using NS2 simulations that was projected on a simulation graph.

Generation of Nodal Space

We use the NSG2.1 software for the creation of TCL file.



```
e: ~
java -jar NSG2.1.jar
```

A map was drawn and nodes projected with 4 TCL receptors

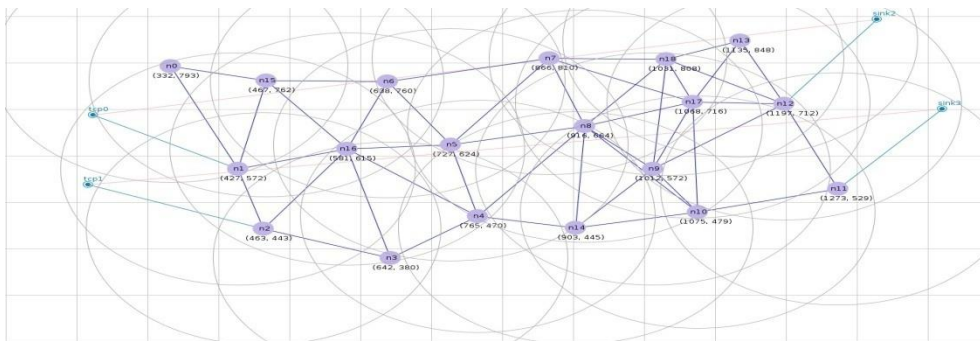


Fig 5.4 AODV.tcl file

- The system was activated and nodes started to interact with each other.
- Output was traced on the nam software.

Using of custom configuration NS2 with TCL to obtain trace files.
Here the code utilized for the running of simulation is shown.

```

# This script is created by NSC2 beta3
# <http://www.hupong.com/pages/centermsg>

#####
# Simulation parameters setup
#####
set val(chan) Channel/WirelessChannel ;# channel type
set val(prop) Propagation/TwoRayGround ;# radio-propagation model
set val(netif) Phy/WirelessPhy ;# network interface type
set val(mac) Mac802_11 ;# MAC type
set val(qt) Queue/DropTail/PriQueue ;# interface queue type
set val(ant) Antenna/OmnisAntenna ;# antenna model
set val(rng) 50 ;# max packet in Rng
set val(nn) 30 ;# number of mobilenodes
set val(pl) AODV ;# routing protocol
set val(x) 1373 ;# X dimension of topography
set val(y) 548 ;# Y dimension of topography
set val(stop) 25.0 ;# time of simulation end

#####
# Initialization
#####
# Create a ns simulator
set ns [new Simulator]

# Setup topography object
set topo [new Topography]
topo load_flatgrid $val(x) $val(y)
ns add-god $val(nn)

# Open the NS trace file
set tracefile [open AODV.tr w]
ns trace-all $tracefile

# Open the NAM trace file
set namfile [open AODV.nam w]
ns namtrace-all $namfile
ns namtrace-all-wireless $namfile $val(x) $val(y)
set chan [new $val(chan)] ;# Create wireless channel

#####
# Mobile node parameter setup
#####
ns node-config -adhocRouting $val(pl) \
    -htype $val(qt) \
    -macType $val(mac) \
    -lqType $val(qt) \
    -lqLen $val(rng) \
    -antType $val(ant) \
    -propType $val(prop) \
    -phyType $val(netif) \
    -channel $chan \
    -topoInstance $topo \
    -agentTrace ON \
    -routerTrace ON \
    -macTrace ON \
    -movementTrace ON

#####
# Nodes Definition
#####

```

(Code given in Appendix)

Algorithm for creating the blackhole simulation:

- In our experiment we are creating black hole nodes which will drop the packet after obtaining them.
- We are implementing the simulations by making changes in the config file of ns2.35 and then re installing the software with those changes.
- At our basic level we are simulating the attack in which our attacker (known) will drop the packets instead of dropping them.
- Here comes the main question, how do we know that the attacker node is actually an attacker, that solution we have given in our journal paper.
- We establish trust value metrics in which there will be 3 trust values of each car on the network.
- High medium and low
- Using fuzzy based approach on the various data and timestamps related to the data which they are transferring we can figure out which is faking its identity which is not.

(with attacker initialization)

```
$n15 set Z_ 0.0
$ns initial_node_pos $n15 20
set n16 [$ns node]
$n16 set X_ 907
$n16 set Y_ 495
$n16 set Z_ 0.0
$ns initial_node_pos $n16 20

# $ns at 1.0 "$n2 set ragent_ blackhole"
# $ns at 1.0 "$n5 set ragent_ blackhole"

#=====
#           Generate movement
#=====
```

(Without attacker initialization)

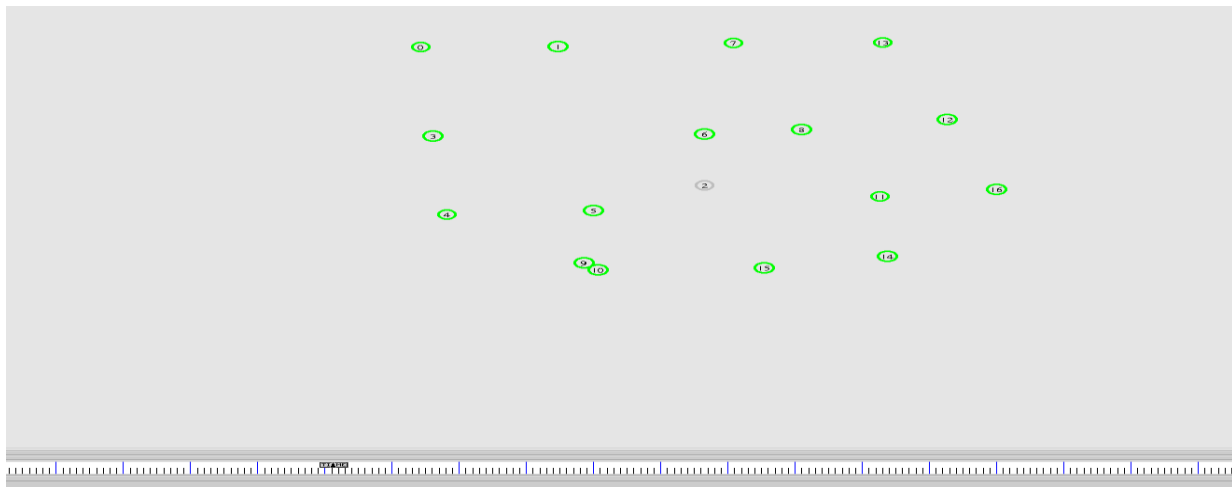



Fig5.5 Simulation of .tcl file



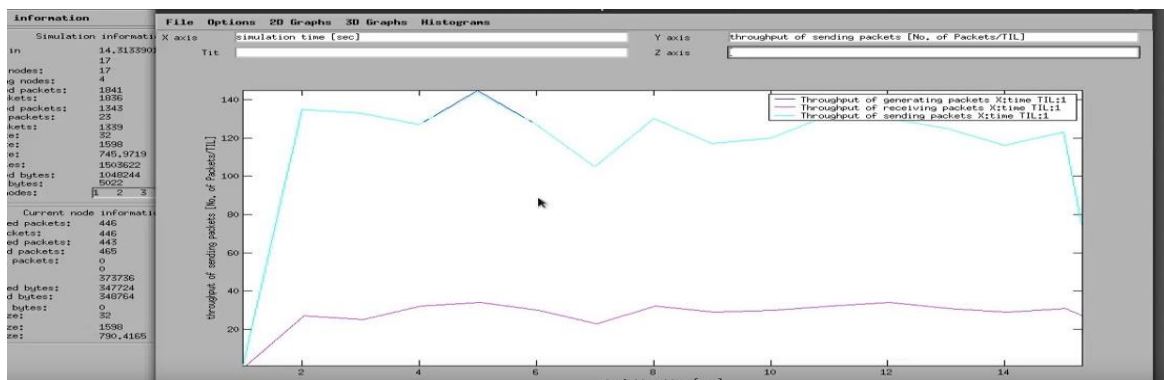
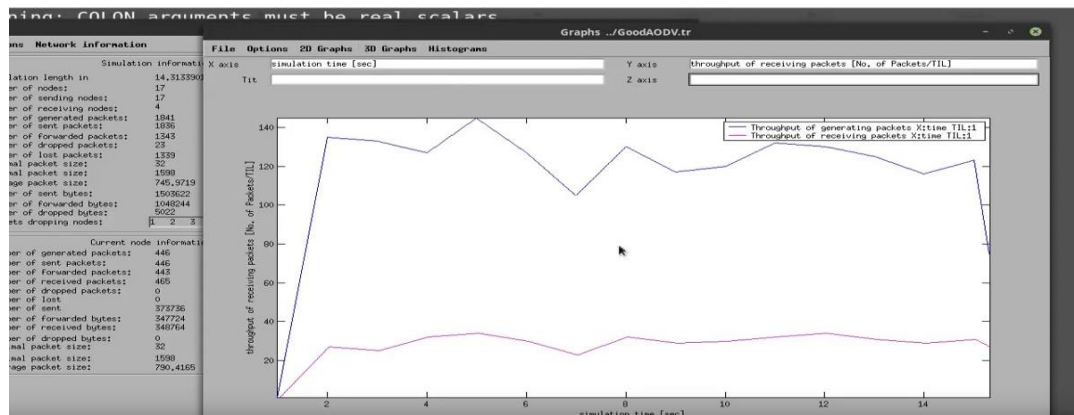
```

N -t 1.029140 -n 3 -e 49.380438
N -t 1.029140 -n 8 -e 49.380101
N -t 1.029140 -n 11 -e 49.379363
N -t 1.029140 -n 14 -e 49.380276
N -t 1.029140 -n 1 -e 49.380191
N -t 1.029140 -n 7 -e 49.380270
N -t 1.029140 -n 0 -e 49.381252
N -t 1.029140 -n 12 -e 49.380032
N -t 1.029140 -n 16 -e 49.380361
N -t 1.029140 -n 13 -e 49.380440
r -t 1.029491568 -Hs 15 -Hd -2 -Nl 15 -Nx 645.00 -Ny 331.00 -Nz 0.00 -Ne 49.379719 -Nl 15
s -t 1.029501568 -Hs 15 -Hd -2 -Nl 15 -Nx 645.00 -Ny 331.00 -Nz 0.00 -Ne 49.379719 -Nl 15
N -t 1.029502 -n 9 -e 49.379882
N -t 1.029502 -n 14 -e 49.380057
N -t 1.029502 -n 10 -e 49.379882
N -t 1.029502 -n 11 -e 49.379144
N -t 1.029502 -n 5 -e 49.379811
N -t 1.029503 -n 6 -e 49.379882
N -t 1.029503 -n 8 -e 49.379882
N -t 1.029503 -n 16 -e 49.380142
N -t 1.029503 -n 2 -e 49.379882
N -t 1.029503 -n 12 -e 49.379813
N -t 1.029503 -n 4 -e 49.380134
N -t 1.029503 -n 3 -e 49.380218
N -t 1.029503 -n 7 -e 49.380051
N -t 1.029503 -n 13 -e 49.380221
N -t 1.029503 -n 1 -e 49.379972
r -t 1.029806321 -Hs 5 -Hd -2 -Nl 5 -Nx 453.00 -Ny 450.00 -Nz 0.00 -Ne 49.379811 -Nl 5
s -t 1.029816321 -Hs 5 -Hd -2 -Nl 5 -Nx 453.00 -Ny 450.00 -Nz 0.00 -Ne 49.379811 -Nl 5
N -t 1.029817 -n 10 -e 49.379394
N -t 1.029817 -n 9 -e 49.379394
N -t 1.029817 -n 2 -e 49.379394
N -t 1.029817 -n 4 -e 49.379646
N -t 1.029817 -n 6 -e 49.379394
N -t 1.029817 -n 15 -e 49.378951
N -t 1.029817 -n 3 -e 49.379731
N -t 1.029817 -n 8 -e 49.379394
N -t 1.029817 -n 11 -e 49.378656
N -t 1.029817 -n 14 -e 49.379569
N -t 1.029817 -n 1 -e 49.379484
N -t 1.029818 -n 7 -e 49.379563
N -t 1.029818 -n 0 -e 49.380575
N -t 1.029818 -n 12 -e 49.379325
N -t 1.029818 -n 16 -e 49.379653
N -t 1.029818 -n 13 -e 49.379733
r -t 1.030505074 -Hs 15 -Hd -2 -Nl 15 -Nx 645.00 -Ny 331.00 -Nz 0.00 -Ne 49.378951 -Nl 15
s -t 1.030515074 -Hs 15 -Hd -2 -Nl 15 -Nx 645.00 -Ny 331.00 -Nz 0.00 -Ne 49.378951 -Nl 15
N -t 1.030516 -n 9 -e 49.379175
N -t 1.030516 -n 14 -e 49.379350
N -t 1.030516 -n 10 -e 49.379175
N -t 1.030516 -n 11 -e 49.378437
N -t 1.030516 -n 5 -e 49.378967
N -t 1.030516 -n 6 -e 49.379174
N -t 1.030516 -n 8 -e 49.379174
N -t 1.030516 -n 16 -e 49.379434
N -t 1.030516 -n 2 -e 49.379174

```

Fig 5.6 Trace File

This is the trace file generated and we use this results to compare the 2 simulations i.e. with blackholes and without blackholes and a graph is plotted to study them using the **xgraph**

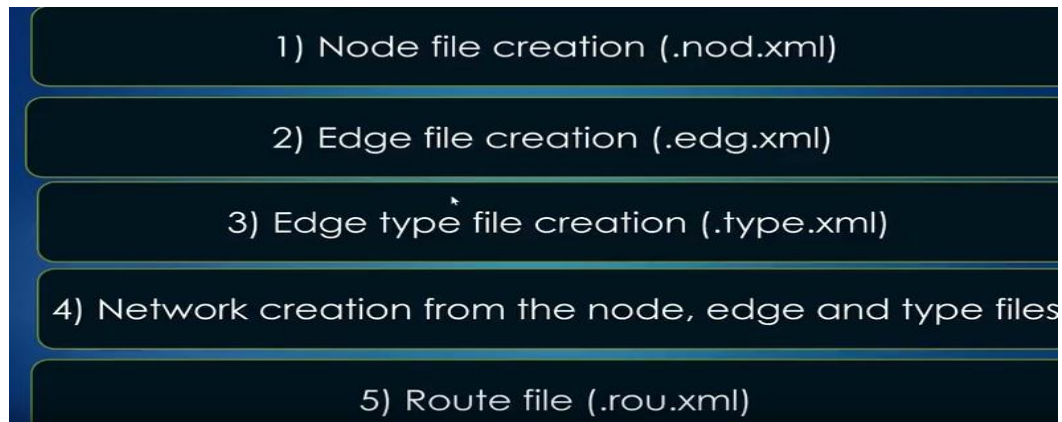


| |
|--------------------------------|
| Number of generated packets: 4 |
| Number of sent packets: 2 |
| Number of forwarded packets: 2 |
| Number of received packets: 22 |
| Number of dropped packets: 16 |
| Number of lost packets: 0 |
| Number of sent bytes: 204 |
| Number of forwarded bytes: 88 |
| Number of received bytes: 1000 |
| Number of dropped bytes: 472 |
| Minimal packet size: 40 |
| Maximal packet size: 102 |
| Average packet size: 49.6923 |

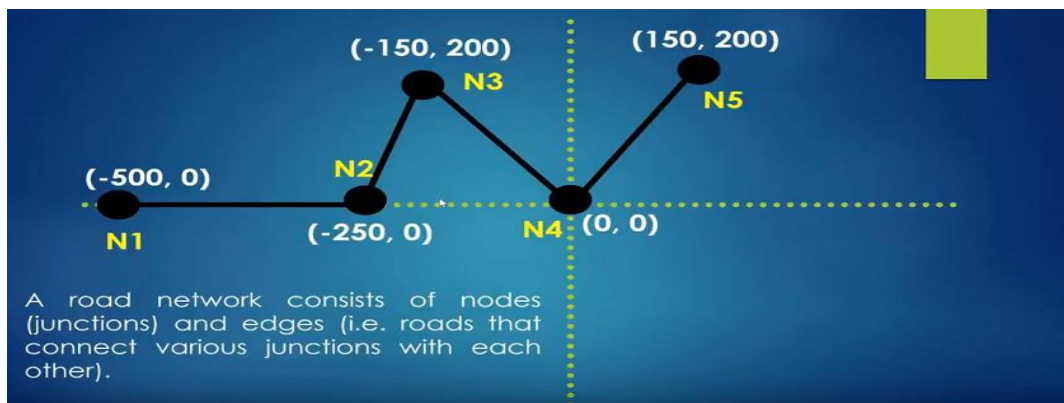
| |
|-----------------------------------|
| Number of generated packets: 446 |
| Number of sent packets: 446 |
| Number of forwarded packets: 443 |
| Number of received packets: 445 |
| Number of dropped packets: 0 |
| Number of lost packets: 0 |
| Number of sent bytes: 373736 |
| Number of forwarded bytes: 347724 |
| Number of received bytes: 348764 |
| Number of dropped bytes: 0 |
| Minimal packet size: 32 |
| Maximal packet size: 1598 |
| Average packet size: 790.4165 |

(Results of the 2 scenarios)

Now running a custom simulation in OMNET++ and SUMO, although ns2 was used to study the attacks this was used to study the protocols of V2V communication.



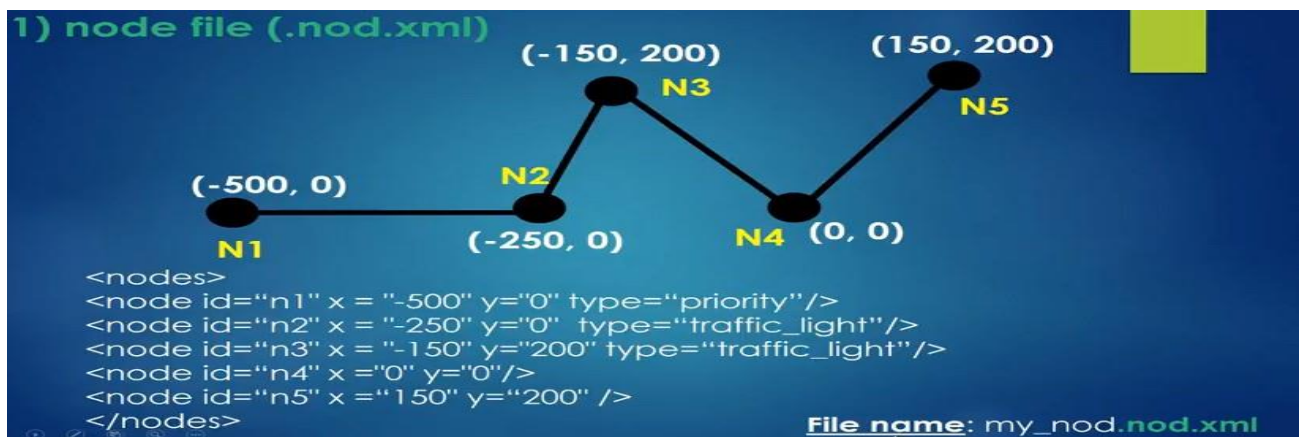
1. Manual route, edge and node assignment



BASIC STEPS

1. Node file creation: create a basic text file and save it as named above.

Each node has a particular node id (unique to it). X and Y are co-ordinates and type is what that node is basically.



2. Edge file creation: Save it as my_edge.edge.xml

n1 to n2 there is a path whose id is 1to2 which is a 3lane road with max speed of 45 km/hr.

And similarly the rest of lines have similar definitions.

2) Edge file (.edg.xml) define the connect node together to form links.

Example:



```
<edges>
<edge from="n1" to="n2" id="1to2" type="3L45"/>
<edge from="n2" to="n3" id="2to3" type="2L15"/>
<edge from="n3" to="n4" id="3to4" type="3L30"/>
<edge from="n4" to="n5" id="out" type="3L30"/>
</edges>
```

Fine name: my_edge.edg

3. Create type file: contains definitions of the types defined in the edge file

3) Type file (.type.xml) include road priority, the number of lanes, speed limit, type of vehicles allow, etc.

Example:

```
<types>
<type id="3L45" priority="3" numLanes="3" speed="45"/>
<type id="2L15" priority="3" numLanes="2" speed="15"/>
<type id="3L30" priority="2" numLanes="3" speed="30"/>
</types>
```

Fine name: my_type.type

Save it as my_type.type.xml

4. Generate the .net.xml file: use the netconvert tool built in SUMO to obtain the .net.xml format of the file.

\$ netconvert --node-files my_nodes.nod.xml --edge-files -t my_type.type.xml -o my_net.net.xml

4) netconvert



```
netconvert --node-files my_nodes.nod.xml -  
-edge-files my_edge.edg.xml -t  
my_type.type.xml -o my_net.net.xml
```

Generate the route file now: save it as my_rou.rou.xml

5) Route file (.rou.xml)

```
<routes>  
<vType accel="1.0" decel="5.0" id="Car" length="2.0" maxSpeed="100.0" sigma="0.0" />  
<vType accel="1.0" decel="5.0" id="Bus" length="12.0" maxSpeed="1.0" sigma="0.0" />  
  
<route id="route0" edges="1to2 2to3"/>  
<vehicle depart="10" id="veh0" route="route0" type="Bus" />  
<route id="route1" edges="2to3 3to4"/>  
<vehicle depart="10" id="veh1" route="route1" type="Car" />  
<route id="route2" edges="3to4 out"/>  
<vehicle depart="30" id="veh2" route="route2" type="Car" />  
</routes>
```

5. Generate a sumo-cfg file now :::

Lastly, the Sumo Configuration file (.sumocfg)

```
<configuration>

<input>
<net-file value="my_net.net.xml"/>
<route-files value="my_route.rou.xml"/>
</input>
<time>
<begin value="0"/>
<end value="2000"/>
</time>

</configuration>
```

File name: my_config_file

Time denotes the simulation time .The net and rou files are those files which you have created to see it that all things are stored in one folder only. Save it as my_config_file.sumocfg

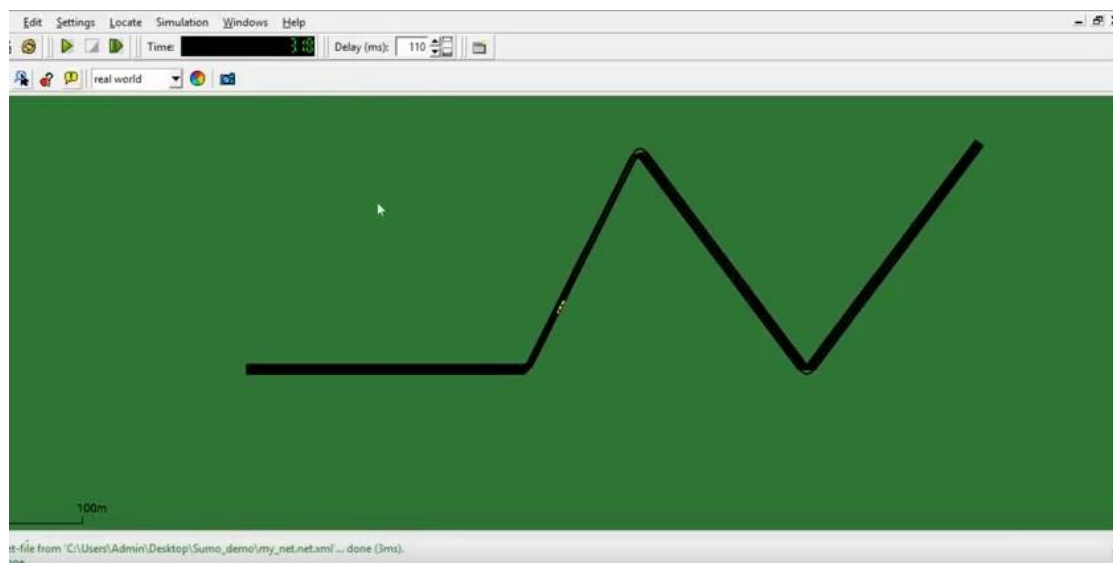


Fig 5.8 SUMO simulation

THEN RUN THE .sumocfg file in sumo-gui , put some delay in the simulation and you will obtain this ::

This is just simple sumo-gui running.

Now,

To implement the Veins module i.e. V2X communication has to be implemented.

Go to the veins module in omnet++ and go to the main **omnetpp.ini** file

```
#####
#           TraCIScenarioManager parameters           #
#####
*.manager.updateInterval = 1s
*.manager.host = "localhost"
*.manager.port = 9999
*.manager.autoShutdown = true
*.manager.launchConfig = xmldoc("erlangen.launchd.xml")
```

In the file: you see the erlangen.launch.xml

In the TraCI manager runs the in-built example erlangen.launch.xml

Go to the system directory where you have this example and u paste there your own my_ files which you had created i.e. the my_node.nod.xml , my_type.type.xml, my_route.rou.xml etc.

Now you have to create your own launch file which points towards the my_ files which you had created

Copy the erlangen.launchd.xml and save it as my.launchd.xml Now, edit the my.launchd.xml file as follow:

```
<?xml version="1.0"?>
<!-- debug config -->
<launch>
    <copy file="erlangen.net.xml" />
    <copy file="erlangen.rou.xml" />
    <copy file="erlangen.poly.xml" />
    <copy file="erlangen.sumo.cfg" type="config" />
</launch>
```

Change it to:

```
<?xml version="1.0"?>
<!-- debug config -->
<launch>
    <copy file="my_net.net.xml" />
    <copy file="my_rou.rou.xml" />

    <copy file="my.sumocfg" type="config" />
</launch>
```

you have to slightly change the config file also:
 **
 copy the erlangen.sumocfg and re-adjust your my.sumocfg accordingly to accompany the
 missing time factors.

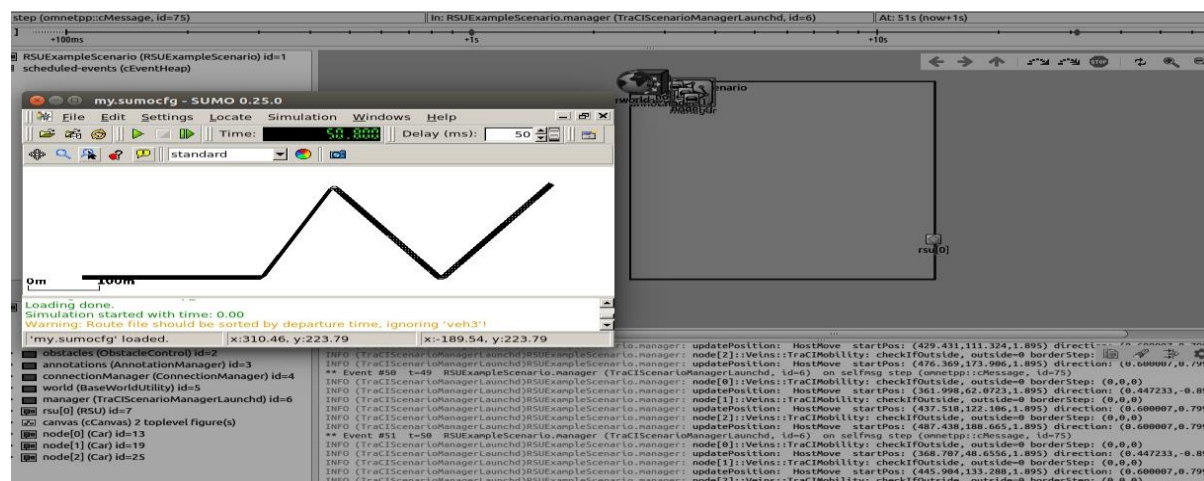
Now in the omnetpp.ini file change it to my.launchd.xml and run the simulation.

\$ /omnetpp-5.3/src/veins-veins-4.7.1\$ python sumo-launchd.py -vv -c /usr/bin/sumo-gui

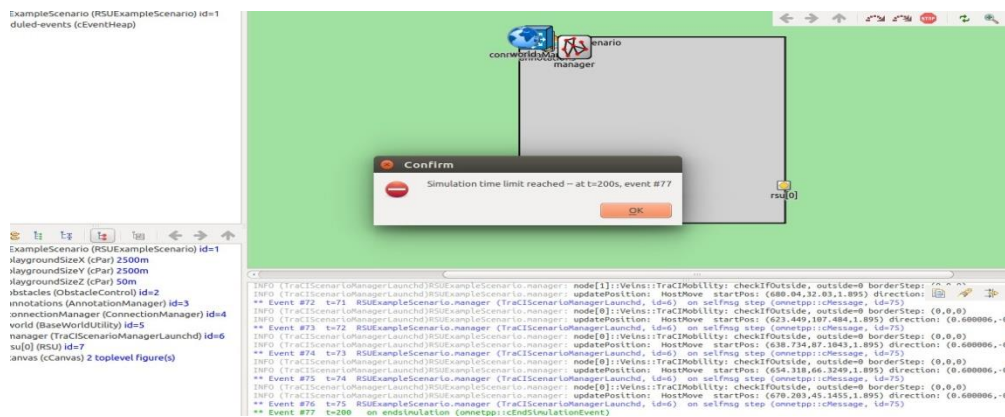
And after the ports start listening run the omnetpp.ini file as **runas omnetsimulation**

And thus you will obtain the following output (produce some delay to view it properly)

Output:



**Fig 5.9 omnet-sumo-simulation
 (PAUSED I.E WHY SCREENSHOT IS BLACK)**



This is a small map:

Now going for custom maps : { actual maps present in our world }

Download the map area from <https://www.openstreetmap.org/>

Select a part, for e.g. a bridge or something

1. Save the file as .osm file and (for eg. guindy.osm)

2. You can directly use this file to produce the .net.xml file

\$ netconvert –osm-files guindy.osm -o guindy.net.xml

3. Now there is a problem, this map contains a lot of other things, thus it's a procedure to generate the poly.xml file of the map which contains additional information about the map.

*** * There is a type file which can be referred for the conversion**

It is present in the SUMO folder i.e. in the sumo-0.32.0 folder from which you installed sumo.

/sumo-0.32.0/data/typemap

copy that map in the directory where you are saving all this .

**\$ polyconvert –osm-files guindy.osm –net-file guindy.net.xml –type-file osmPolyconvert.typ.xml
-o guindy.poly.xml**

4. Now to generate the rou.xml file which is generated with the help of a python script which generates random trips across the map.

cd to sumo-0.32.0/tools to randomtrips.py

Copy the .py file to the folder where you are constructing all your other files.

*** now run the .py script with the above created .net.xml file as input and output as .rou.xml file**

\$ python randomTrips.py -n guindy.net.xml -r guindy.rou.xml -e 50 -l

-50 refers to 50 cars

This will generate the guindy.rou.xml file

now you have obtained all the required files...now create a .sumocfg file by copying the eg.

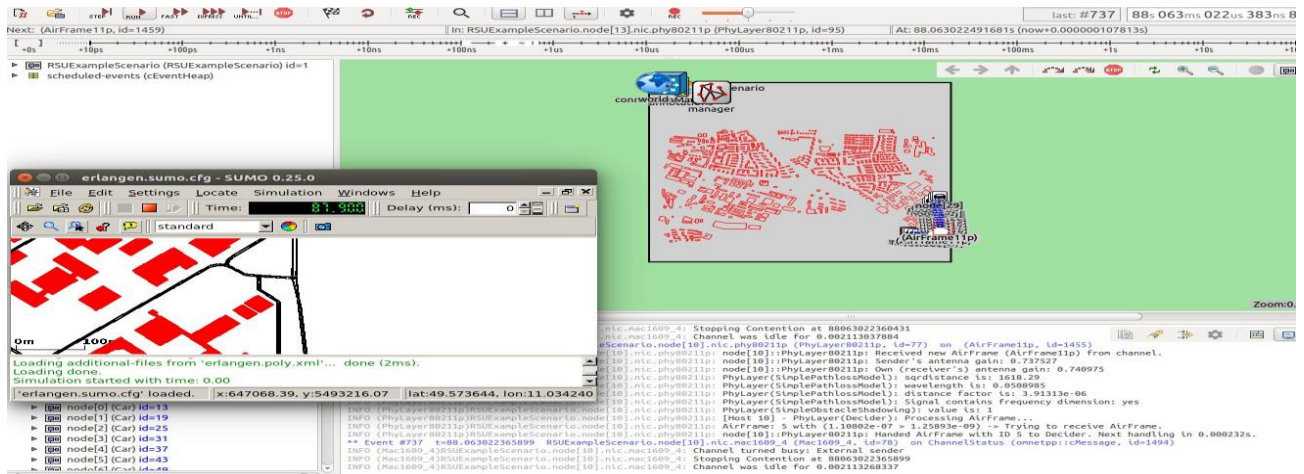
Erlangen.sumocfg file and changing certain parameters.

Similarly copy the erlangen.launchd.xml file and make the desired changes.

NOTE:::this changes are basically made to point to the correct .net.xml .type.xml etc files.

NOW SIMILARLY RUN THE SCRIPT as the above eg . i.e the sumo-launchd.py from the terminal and get the port to listen to the output of the omnetpp.ini file .

NOTE::::in running big maps which are all custom made ::: try to change the parameters accord-



ingly for errors ::: for eg. If it shows that the custom map has some part OUTFBOUNDS
change the size from the **omnetpp.ini** ie the simulation parameters

You will face similar errors many times but make changes accordingly to the error displayed as the
omnetpp.ini file was originally configured for the inbuilt example.

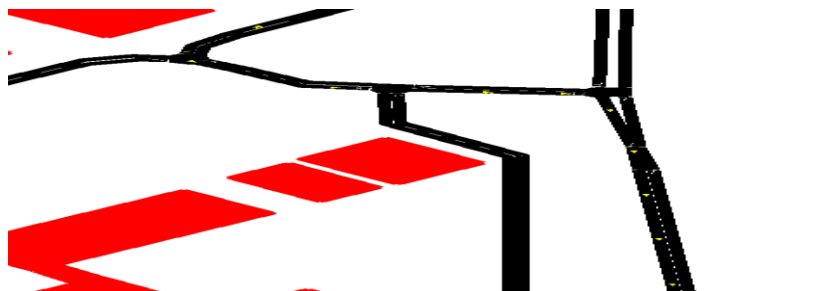


Fig 5.10 V2V-omnet-simulation

The deployed vehicles are communicating with each other.

Fig 5.11 Simulation

You can see the nodes are deployed and communicating with each other.

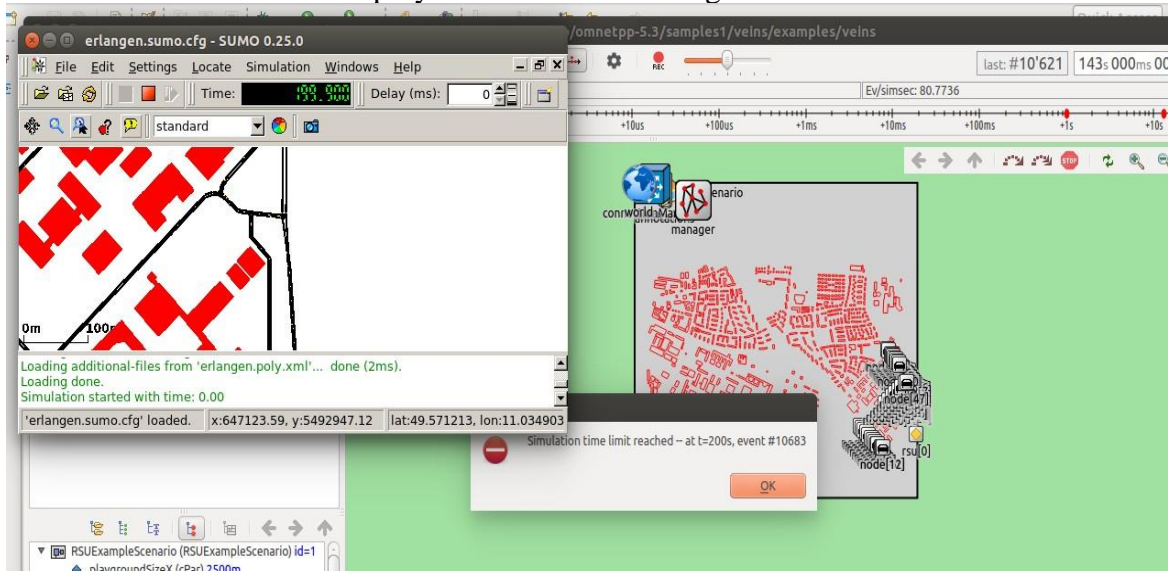


Fig 5.12 End of simulation.

```

#733      88.063022231713  node[11] --> node[14]      1461      22003      144 bytes
#733      88.063022231713  node[11] --> node[16]      1465      22003      144 bytes
#733      88.063022231713  node[11] --> node[17]      1467      22003      144 bytes
#733      88.063022231713  node[11] --> node[18]      1469      22003      144 bytes
#733      88.063022231713  node[11] --> node[19]      1471      22003      144 bytes
#733      88.063022231713  node[11] --> node[20]      1473      22003      144 bytes
#733      88.063022231713  node[11] --> node[21]      1475      22003      144 bytes
#733      88.063022231713  node[11] --> node[22]      1477      22003      144 bytes
#733      88.063022231713  node[11] --> node[23]      1479      22003      144 bytes
#733      88.063022231713  node[11] --> node[24]      1481      22003      144 bytes
#733      88.063022231713  node[11] --> node[25]      1483      22003      144 bytes
#733      88.063022231713  node[11] --> node[26]      1485      22003      144 bytes
#733      88.063022231713  node[11] --> node[27]      1487      22003      144 bytes
#733      88.063022231713  node[11] --> node[28]      1489      22003      144 bytes
#733      88.063022231713  node[11] --> node[29]      1492      22003      144 bytes

```

Fig 5.13 Log files of communication

Adjust the playground Size to avoid simulation-run-time-errors.

By performing various simulations we can effectively study the V2V simulations and find how to repair the flaws in the system.

Chapter 6

MITIGATION OF THE ATTACKS

The only solution for this is to maintain the privacy of the driver, there have been a lot of cures developed to overall maintain the privacy of the driver within the network, the true identity of the vehicle is kept hidden with the use of a set of anonymous keys which are constantly or variably changing with time according to the driving speed of the vehicle or through the use of pseudonyms [15] or either using asset of group signatures should also be used as mentioned in [11][13][14], which are some certain ways to maintain privacy. The researchers in [3] suggests the pre-loading of anonymous keys in the TPD (tamper proof device) which are recognized by the regional certificate authority and can be detected back to Electronic License Plate authorities (ELP). At the very basic idea we have to maintain a pseudo information which cannot be tracked and should be changing according to the situations i.e. should use the current driving parameters as input to create different keys so that one can maintain his identity with the system but also not get recognized by the attackers

DOS Attack

It can be decreased to some extent by using the digital signature method [16], to ensure secure and reliable message communication and authentication Digital signatures are used. It is an active security measure that is attained by digitally signing the data [1], all in all it takes time for the attacker to penetrate a system running on customized hardware which is using non- public protocols. In Dos, DDos attacks the major solutions aside from adding different digital signatures and symmetric keys would be to identify the attacker based on certain algorithms which could rule out the attacker from the network, which is shown through a trust model proposed in [17] that ensures trustworthiness of the vehicle by calculating, the trust metric values of nodes participating in VANET. By calculating the number of accepted and received messages it calculates the trust value which is used to rule out the attacker from the network through the help of a fuzzy based approach to identify malicious nodes, and this information is utilized by sending a direct request to the RSU or CA and is driven to action when the threshold (which is the case of DDoS, Dos attacks) is exceeded.

Malware and Spamming:

The only way to avoid malware implantation would be to have digital signatures involved in every hardware and software part of your car and to get it updated/checked on regular bases. Avoid the use of non-trusted/pirated/customized software, only use trusted hardware as it makes it very difficult to replace the existing protocols and values, monitor data excluding authorized nodes. Manufacturing companies need to regularly check the software being installed on the car for vulnerabilities, and have a systemized check on the employees to avoid malicious employees adding malware to the vehicle.

Sybil attacks

Sybil attack basically forges a vehicle identity in multiple places, the only way to avoid this is to maintain a TFD – database which will store node information and rule out identical nodes, u can use the location attribute to rule out the fake nodes on the network, in [3], Sybil attacks are prevented by making use of location and various other information of the transmitting node and coming to a logical conclusion whether the information presented by it is logically correct or not. Messages are received by vehicles and corresponding certificate are examined along with its, life time and location. If it is correct logically then the vehicle accepts the message, or else it informs the nearest certificate authority. The other way is to make a temporary changing certificate system which would give the authenticity of a node i.e. A Central Validation Authority is deployed (VA), which validates entities in real time directly or indirectly using temporary Certificates [18].

Man in the Middle Attacks

The only way to avoid others from listening to the v2v communication or even letting them access the tunnel for v2v communication would be to use strong authentication methods like digital certificates which use strong digital signatures [7] and establishing connections between nodes via secure keys or encrypting the data being transmitted.

Attacks pertaining to Hardware and Software

Message tampering

There are many ways, one way to avoid message tampering would be to use some form of group signatures [13], which not only helps to maintain your privacy but also helps in maintaining the authenticity, integrity, and accountability of the nodes on the network in which tampered mes-

sages for unauthorized node are detected by the use of probability-based signature verification scheme/algorithm, such a method is usually called data correlation. Other way would be to use the similarity algorithm [20] which propose to use a trust management framework and a reputation management framework whose values are calculated via the similarity algorithm and trust of messages content value between vehicles to help driver to believe or not to believe the received message. By calculating the trust value if it surpasses a threshold they take appropriate action and rebroadcast the message, Otherwise they drop it and report the activity to the CA or theRSU.

For spoofing attacks

One of the quintessential ways to prevent the spoofing attacks, would be to maintain the identity of the vehicles in the network and removing the nodes that project a malicious identity as we have discussed above but if at the end the message is somehow being tampered and deployed in the network then you need to use the advanced form of vehicular PKI (VPKI) as used by the EU and VSC-A for authenticating the processes and message exchange between vehicles. VPKI possesses a group of trusted third parties such as, one certificate authority in each city/state or country, with authorized certificate authorities in the particular area, certificate authorities act mutually and can be recognized by the vehicles in different countries or areas with the help of data sharing among them. A set of private and public keys are issued for each and every vehicle along with the use of temporary or short-time based certificates with anonymous keys which are continuously changing according to the driver's speed [19][15]. Electronic License Plates on the vehicles and their corresponding pseudonyms can only be differentiated by legal authorities so a circulated signed message along with its certificate is authenticated via a certificate authority and is legislated as vehicles passes through a certain route. Hence a secure channel of communication is established between authenticated users.

Resistance offered against routing attacks (Black hole, Greyhole and Wormhole):

Soft wares and the sensors make us of the digital signatures to avoid data tampering or being accessed by the attacker, and help to rule out fake messages being broadcasted on the network. In ARAN (Authenticated Routing for Ad-hoc network and SEAD (Secure and Efficient Ad-hoc Distance Vector routing protocol cryptographic) [2],symmetric cryptography cryptographic certificate, one way hash function and MAC (Message Authentication Code) are used

respectively to solve these issues which involve at the very core spoofing identity to get access to the message on the network. In [7], an effective technique is discussed to tackle the Wormhole attack and the various other routing attacks in the network with the understanding of HEAP. It is a protocol based on the AODV algorithm and makes use of geographical criteria to limit the travelled distance from source to destination; if the limit is surpassed then the data/packet is dropped. The HEAP protocol basically works on ruling out improper locations or spoofed locations and is mostly used to stop Sybil attacks and similarly uses a fuzzy based approach to calculate the authenticity of the message from the various parameters of the message.

Attacks on the vehicular sensors

Against jamming attacks

The only solution to this problem would be to switch between the available wireless technologies in the vehicle to continue monitoring the environment if one type of sensors are blocked like suggested by the authors in [21].

Attacks against the Physical Infrastructure

For unauthorized access

At the very core, the only way to avoid unauthorized access to CAs and RSUs network would be by implementing proper identification of the individual (but also maintaining the drivers privacy at the same time) through the help of TFD and the pseudo keys in it which can help authenticate users in the network and administrators separately at the same time [3] and also by implementing proper digital certificates with digital signatures and cryptographic symmetric methods to encrypt the network so that only authorized users with the correct digital signatures can access it. The other part of this problem is that instead of virtually attacking the network they try to physically attack it by visiting the service in the real world and tampering with it which can only be avoided by implementing proper security officials to maintain a watch over the area for illegal access to the facilities of CAs and OBUs.

Chapter 7

SOFTWARE REQUIREMENTS

- ITS is implemented (simulated) using ++OMNET (via the VEINS package) and SUMO.
- SUMO is an open source, exceptionally compact, tiny and consistent street traffic reproduction bundle intended to deal with expansive street systems. It is for the most part created by workers of the Institute of Transportation Systems at the German Aerospace Center. SUMO is open source, authorized under the GPL.
- OMNET++ is a particular, segment based C++ reproduction library and system, basically for structure arrange test systems. SUMO is an open source, very convenient, minute and ceaseless street traffic reproduction bundle intended to deal with extensive street systems. It is for the most part created by representatives of the Institute of Transportation Systems at the German Aerospace Center. SUMO is open source, authorized under the GPL.
- Veins is an open source system for running vehicular system reenactments. It depends on two settled test systems: OMNET ++, an occasion based system test system, and SUMO, a street traffic test system.

TOOLS:

- openstreetmap.org for downloading maps in xml format
- SUMO 0.32.0 or 0.28.0
- veins
- omnet++ 5.1
- NS-2 (2.35-allinonens)
- NAM
- Xgraph
- NSG 2.1

OS

- Linux Ubuntu 16.04
- python and python3 for compilations , py-bind

Chapter 8

CONCLUSION

With the advancement in transportation technology the risks are also increasing with it, stats show that there had been more than 11 million accidents this year and people die or get involved in accidents more now, this shows that although we have developed the technology but have not made it more secure as (the increase in accidents per year show that).Due to increasing number of accidents on the roads users want safety and security on the most common medium of travel and a strict procedure to check the misbehavior and malicious activities of others that may cause harm to fellow travelers and also to the collateral as a whole. Certain preventive measures are to be enabled in near future and more efforts are undertaken by authorities to pertain a secure VANET environment.

Chapter 9

FUTURE ENHANCEMENTS

In this project certain attacks were studied and simulated and with the advancement in the current technology there are bound to be many security flaws, thus the future work which can be done at the core level would be to study the new types of network attacks and figure out techniques to mitigate them by understanding their working via a simulation environment.

