VANET Simulator: Full Design Architecture

Mohamed M. A. Elgazzar, Ahmad Alshareef

Abstract: ***Vehicular ad-hoc networks VANETs are now greatly affecting our daily lives. Autonomous driving vehicles becomes a hot research topic in automotive industry. Deploying autonomous vehicles scenarios requires realistic network through which all vehicles can communicate safely and efficiently. Network simulation prior direct implementation saves a lot of time and money, due to the high implementation cost of the vehicular ad-hoc networks.VANET simulators are not as same as mobile ad-hoc network MANET simulators; they have different characteristics and different architecture design, where VANET simulators are rapidly changing network topology in a frequent basis with high-speed moving nodes.In this paper, we will discuss the full architecture design of VANET simulator, which consists of three main parts: mobility generator, network simulator, and a middleware framework that combine both mobility and network simulator together, followed by a comparative study of the different parts of VANET simulator.***

Keywords: VANET, MANET, Mobility generator, Network simulator, Frameworks, Simulation, Emulation, Federated and Integrated approach.

I. INTRODUCTION

Network simulation step is very important step in the full development of constructing any kind of networks as it helps the designer to implement his work directly, saving time and money. Network simulation saves time as it is a speedy way to implement the network with no need to connect the hardware to the network, and it saves money as real simulation of a network is expensive and is not flexible to any changes needed to be implemented in the network. Simulation is an important part in many fields of science like physics, chemistry and for sure engineering, not only in computer engineering but also in civil engineering and other fields of engineering as well. Simulation in networking helps the designer to implement his protocol or algorithm and to prove the concepts behind his algorithm or protocol. Simulation focus more on modeling the network and calculate the behavior of the network in virtual mode of simulation, as its results are not build on real scenarios, but a simulated scenarios that are cheap and fast to implement. Researchers, developers, and quality assurance engineers use simulation in order to design, simulate, analyze and verify the proposed protocol or algorithm.

1.1. VANET vs. MANET Simulation

Vehicular ad-hoc networks (VANET) simulation is fundamentally different from mobile ad-hoc networks (MANET) as VANETs have their specific requirements and applications like safety and warning applications.

Revised Manuscript Received on February 05, 2020.
* Correspondence Author
Mohamed M. A. Elgazzar, Faculty, Department of Technological Industry and Energy, Delta Technological University, Egypt., E-mail:m_elgazzar1961@yahoo.com,
Ahmad alshareef, Faculty, Department of Engineering , Cairo University, Giza, Egypt, E-mail:ahmad.alshareef.89@gmail.com

Time critical applications, where end-to-end delay should be highly considered, Should be modeled as VANET networks since any latency in the arrival time of messages may lead to an accident, not likely MANETs that do not include rapidly change topologies networks. Despite vehicular ad-hoc networks (VANET) are considered as MANET network, but it is a special kind of MANETs as it has some different characteristics than MANETs like rapidly change network topology that is very frequent and also very fast. Another important factor is the mobility patterns that are high due to the high speed of movements of nodes compared to the nodes of MANET networks.
Mainly the characteristics of VANETs is different from the corresponding MANETs networks in the following points [1]:

(1) On board sensors: in order to provide the network with the required data, nodes should be equipped with sensors, and GPS module to provide the network with the location of the node.
(2) Patterned mobility: mobility pattern depend on the type of road in rural, urban and semi urban.
(3) High dynamic topology: Vehicles moves at high speed depending on the mobility pattern and hence changes their position rapidly, so the topology used in VANET networks is changes in a dynamic way.
(4) Unlimited battery power and storage: Vehicles have unlimited battery, so the power consumption aspects are not big challenge unlike in other ad-hoc networks.
(5) Frequently disconnected network (intermittent connectivity): Vehicles are moving in high speeds and in different directions, so the link between two nodes can easily disconnect and hence the node can be easily lost from the network. Robust routing protocols should be used in order to overcome this expected disconnection of nodes.

Table 1: Comparison between MANETS and VANETs

	MANET	VANET
Production cost	Low cost	High cost
Change in Network topology	Sluggish and slow	Frequent and very fast
Mobility	Low	High
Density in node	sparse	Frequent variable and dense
Bandwidth	Hundred kps	Thousand kps
Range	Up to 100 m	Up to 600 m
Life time of nodes	Power source	Vehicle lifetime
Reliability	Medium	High
Moving pattern	Random	Regular

II. VEHICULAR AD-HOC NETWORKS DESIGN ARCHITECTURE

Deploying VANET networks and testing their behavior involves high cost and intensive work, therefore simulation is very important and useful prior to direct implementation of the networks.Based on the mentioned characteristics of vehicular ad-hoc networks (VANETs) and the difference from MANETs, the existing VANET simulation software is classified into three different categories [2]; vehicular mobility generators, network simulators and middleware frameworks as shown in Figure 1.

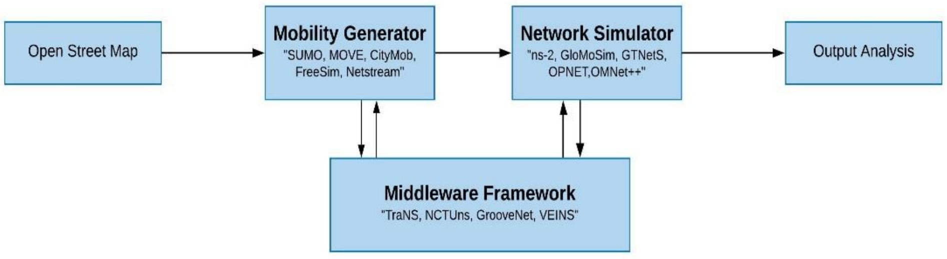

Open Street Map
Mobility Generator
"SUMO, MOVE, CityMob, FreeSim, Netstream"
Network Simulator
"ns-2, GloMoSim, GTNetS, OPNET,OMNet++"
Output Analysis
Middleware Framework
"TraNS, NCTUns, GrooveNet, VEINS"

Figure 1: Vehicular ad-hoc network simulation software

Vehicular mobility generators are required in order to increase the reality level of the simulation done in VANET simulators, they generate realistic vehicular mobility traces that would be treated as input to the network simulator. The provided traces generated from vehicular mobility contains details of the location of each node of the network at every time instant of the simulation including the mobility profiles. The information provided to the network simulators is very important in terms of the realistic output from the simulator at the end. Inputs to the mobility generator includes the type of the road and the scenario factors and parameters like the rate of arrivals and departure, maximum and minimum speed velocity of the nodes in the network and other parameters to be used in the mobility generators.

Many mobility generators used in the VANET simulators like SUMO, CityMob, FreeSim, Netstream, MOVE and others. Comparison between these mobility generators is discussed later on this paper.

Network simulators perform the basic tasks of the simulator as performing packet level simulation of source and destination, transmission and reception of data traffic. Many network simulators used in the VANET simulators like ns-2, GloMoSim, SNS, OPNET, OMNet++ and others. Comparison between these network simulators is discussed later on this paper.

Middleware framework integrate between both network simulation and traffic flow simulation. It is responsible for handling the communication between them by building the two-way link between the two simulators that are simulating in parallel manner. Many vehicular ad-hoc networks frameworks used in the VANET simulators like TraNS, GrooveNet, Veins and others. Comparison between these middleware frameworks is discussed later on this paper.

III. VEHICULAR MOBILITY GENERATOR

1.2. Vehicular mobility modeling

Vehicular mobility generators are required in order to increase the reality level of the simulation done in VANET simulators. The mobility models used in modeling mobility generators are various and have many approaches in building the mobility generators. In order to be more realistic, the mobility model should consider the following factors [3]:

(1)Accurate and realistic topological maps: the model should consider real maps that contain different types of streets with varying densities of vehicles and varying velocities of the network nodes.

(2)Vehicles acceleration and deceleration: vehicles of the network do not suddenly stop and move, so acceleration and deceleration should play important role.

(3)Obstacles: mobility and wireless communication obstacles.

(4)Attraction points: some points are the final destination of many drivers that may cause bottlenecks at this attraction points, we can consider that drivers drive from repulsion points to the attraction ones.

(5)Weather and traffic conditions: climate conditions play important role, also the traffic conditions like rush hours, holidays and going to and from schools.

(6)Driving patterns: driver interact with not only the static obstacles but also with the dynamic obstacles as neighbor vehicles and pedestrians, so the mobility model should be capable of handling all surrounding situations that affects in the network.

3.1.1 Vehicular mobility modeling approaches

Vehicular mobility modeling has many approaches to build mobility generator for vehicular ad-hoc networks (VANETs), the approaches used are the following [4]:

(1)Random models: random mobility patterns are widely used in telecommunication and computer science fields due to their stochastic properties of the distribution of the nodes in the network. These patterns are not suitable for VANETs as vehicles in vehicular ad-hoc networks are not moving in random way, they are moving in known trajectories and roads.

(2) Flow models: Considering vehicles interaction behavior with neighbors and with the surrounding environment, modeling the mobility generator as flow is suitable for VANETs. There are three main types of flow models, macroscopic, mesoscopic and microscopic modeling.

(3)Traffic models: Considering all surroundings factors that affect the traffic like the road traffic red and green light and the global path that the vehicle follows.

(3) Behavioral models: behavioral models depend on the studied behaviors of the driver through artificial intelligence by introducing learning procedures to study the behavior of the driver, as the driver is human not robot.

(4) Trace and survey models: traces from network simulators can be helpful in order to model the mobility model based on the provided traces, also survey of drivers behavior is another source of information that help in constructing the mobility model. The absence of traces in some cases is the big challenge of this approach of modeling.

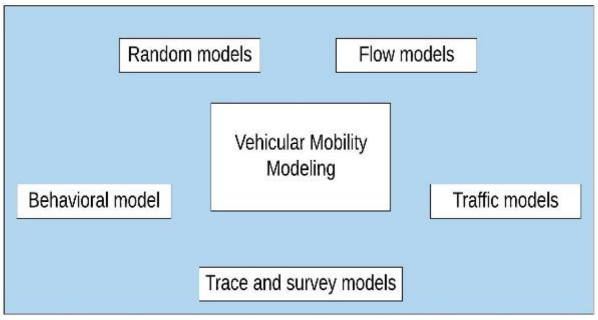


Figure 2: Vehicular mobility modeling approaches

1.3. Vehicular flow model classification

Flow models are classified into three main types; macroscopic, mesoscopic and microscopic modeling.Macroscopic models, model traffic at large scale considering the model as liquid applying hydrodynamic flow theory to the behavior of the vehicles. Macroscopic models do not need high memory requirements in computing, as they do not monitor the network in details like other approaches. Microscopic models, model the behavior of each single vehicle of the network and the interaction of this vehicle with other vehicles of the network, hence microscopic modeling is the most suitable approach to model VANETs. Mesoscopic models, combine both macroscopic and microscopic characteristics, but not to the same level of microscopic models. Vehicular mobility modeling consider both macroscopic and microscopic modeling while modeling VANETs, where macroscopic mobility refers to the big picture of the network including all macroscopic aspects like road topology, speed limit per each road, safety rules per each road, traffic signs descriptions and other aspects. While the microscopic mobility refers to the detailed picture per each vehicle including driver individual behavior, interaction of driver to static and dynamics obstacles, speed acceleration and deceleration and all general driving attitudes including the sex, age, mood and other characteristics of the driver. Therefore, it is preferable for VANET simulator to include both macroscopic and microscopic mobility models when modeling vehicular movement.

3.2 Existing Vehicular mobility generators

Vehicular mobility generators are used in VANETs simulation in order to provide the network simulator with realistic movement traces of all motions of the vehicles of the network, to improve the realistic scenarios of the simulations. Many mobility generators are used to perform this task of providing such traces. We will discuss some of these mobility generators in this section.

A. SUMO

Simulation of urban mobility (SUMO) is collision free, continuous, microscopic and multi modal traffic simulation that was proposed as an open source mobility generator to help in simulating VANET networks and provide the network simulators with more realistic traces, to model the network and analyze it before the implementation phase of the designed network. SUMO is capable of modeling large cities mobility networks with one million vehicles included in that network. As SUMO is multi modal traffic simulators [5], it can model not only the original vehicles but also it can model public transportation vehicles as well. Traffic flow simulation is in microscopic model, which means that the simulator generate traces including detailed information for each vehicle individually, including their position and speed. SUMO is developed in C++ language and has its own GUI to make the dealing with the tool easier. SUMO generated traces are not compatible with available network simulators like NS-2 or QualNet, which is a serious shortcoming.

B. MOVE

Mobility model generator for vehicular networks (MOVE) is mobility generator that is built on top of SUMO simulator in order to generate realistic mobility traces that is suitable to be input to network simulators like NS-2 and QualNet. MOVE is implemented in Java and consists of two main components [6]:

(1) Map editor: used to create road topology. The maps can be created manually or automatically or directly imported from Google earth.

(2) Vehicle movement editor: help user to define the routes of the trips that the vehicles in the networks will take. It defines not only the routes but as well other properties of the route like number of vehicles inside the route, departure time of the vehicle, destination of the vehicle, the duration of the trip and other properties.

C. FreeSim

FreeSim mobility generator is a freeway traffic simulator that models free flowing transportation systems as weighted directed graphs determined by the current speed [7]. FreeSim is both macroscopic and microscopic mobility simulator that model the macroscopic part of the network and the individual details as well, where vehicles can communicate with the simulator to get feedback of the traffic.

The graphical user interface opened in a browser with Adobe flash 8 plug-in, it is an open source simulator licensed by GNU. FreeSim allows shortest path and fastest route, where it applies different approaches at start up or upon request, to provide the vehicle with the requested shortest path or fastest route.

D. CityMob

CityMob mobility generator is an open source simulator that allows user to create urban mobility scenarios including the possibility to simulate accidents and to use flooding based broadcasting events. It generates mobility traces of the vehicles that are compatible with NS-2 network simulator, its implemented using C programming under GNU license.
CityMob propose three different mobility models [8]:
(1)Simple model (SM): Vertical and horizontal mobility patterns are allowed without supporting semaphores.
(2)Manhattan model (MM): city is modeled as Manhattan style grid, where all streets are two ways and car movements are constrained by lanes. It support semaphores at random positions. When a semaphore blocks a vehicle, it remain stopped until the semaphore turns green.
(3)Downtown model (DM): Traffic density is added to Manhattan model to add more realistic to the traces. DM is the best mobility model in CityMob.

3.3 Qualitative comparison between different vehicular mobility generators

To summarize the properties of different mobility generator, Table 2 is built up to compare between these simulators.

Table 2: Qualitative comparison of studied mobility simulators

	SUMO	MOVE	FreeSim	CityMob
Open source	T	T	T	T
Portability	T	T	T	T
GUI	T	T	T	T
Continuous development	T	F	-	T
Real maps	T	T	T	F
User-defined maps	T	T	F	F
Macroscopic	F	F	T	F
Microscopic	T	T	T	T
NS-2 support	F	T	F	T
QualNet support	F	T	F	F

IV. NETWORK SIMULATORS

4.1Simulation vs Emulation

Simulation and emulation are not the same, where simulation [10] is a state of implementing the network through which the performance of this network is evaluated. Emulation is to emulate the implemented network by connecting the implemented network to real hardware and studying its actual behavior and internal behavior of this hardware. The emulator is slower than the simulator as it consider the inner behavior of hardware, but simulators models this hardware with no real case effect.

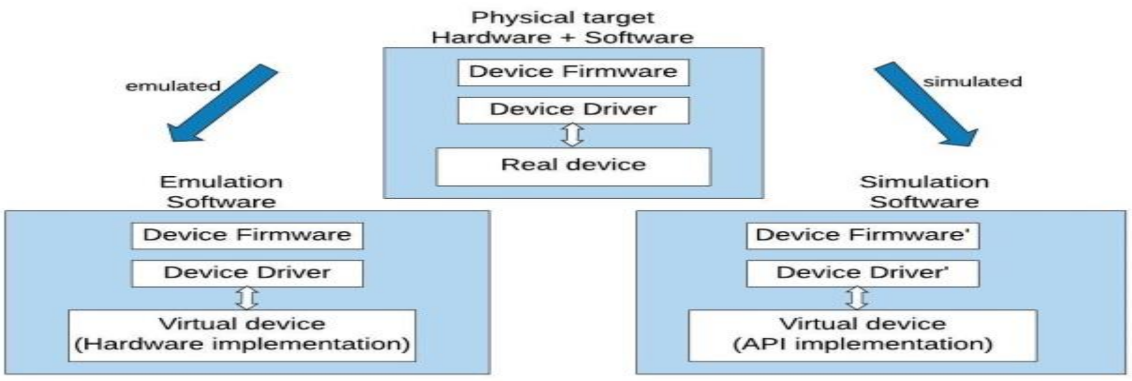


Figure 3: Simulators vs Emulators

The output of the simulators is the trace files of the simulation that log every packet and event happened in the simulation, these traces are used to analyze and study the behavior of the network by the developer or quality assurance person. The generated traces are propagated to other tools that are considering the visualization of such traces to be easier in the analysis.

4.2 Types of network simulators

There are different types of simulators [9], Commercial and open source, where the source code of the commercial software is not free and licensed by the Owner Company or group, while open source simulators are free to use by researchers and user with no charge of using the simulator software. Commercial simulators have advantage over the open source ones, as they have some specialists who are working in a dedicated way to fix any bugs in this simulation, and provides continuous support for the users including the available documentations of the simulator. Open source simulators have their own advantages, as they are flexible to reflect new and recent technologies and topologies in a very fast way than the licensed simulators, but they are poor in documentation and support. Other types of network simulators are the simple and complex simulators, where it all depend on the presence and ease of use the graphical user interfaces (GUI) of the simulator and the command line interface (CLI).

4.3 Existing network simulators

Network simulators are used in vehicular ad-hoc networks (VANET) to implement and simulate the proposed network with the provided traces from the vehicular mobility generator in order to get results that are more realistic which helps in the analysis the behavior of the network by applying any protocol or algorithm.Many network simulators are used to perform this task of simulating the vehicular ad-hoc networks. We will discuss some of these network simulators in this section.

A. NS-2

Network simulator version 2 (NS-2) is event driven network simulator that was developed by VINT research group in the University of California in Berkeley. It was extended by Monarch research group in University of Carnegie Mellon to include extra capabilities like node mobility, realistic physical layer including radio propagation layer, radio networks interfaces, and IEEE 802.11 MAC protocol with distributed coordination function (DCF). Several versions of NS-2 have been released including many extensions for the simulator.
The inputs to NS-2 simulators are TCL simulation scripts that is the input argument to the network, while the outputs of NS-2 simulators are either NAM files or traces.

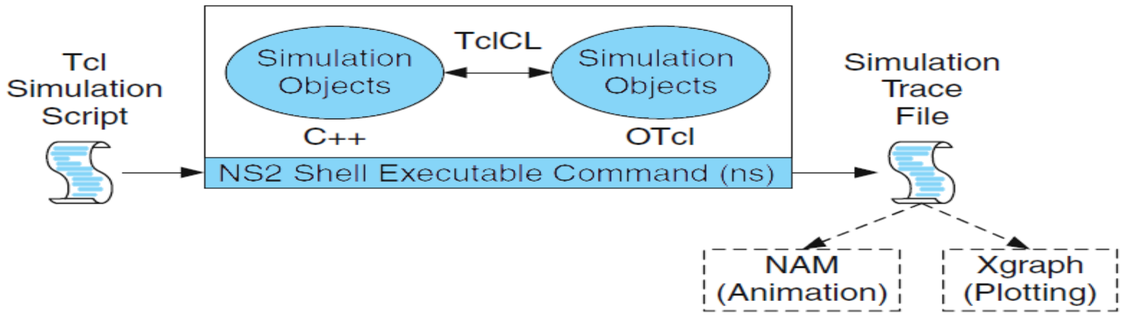


Figure 4: Basic architecture of NS-2 simulator

Network simulator version 2 (NS-2) [11] uses two different languages first is C++ and the second is object oriented tool command line (OTCL), where C++ defines the internal structure and mechanism of the objects of the simulation, while the OTCL defines the external environment of the simulation for configuring the objects.
NS-2 distribution code has some shortcomings in the architecture and in IEEE 802.11 MAC and physical modules. Modifications on the architecture and the two modules are done on the simulator resulting in physical PHY layer that can support any single frame channel communication network, and MAC layer of CSMA/CA mechanism that is required in credible studies.
NS-2 has some limitations in working on windows as it requires in the installation of the software, software program to create UNIX like environment, and this software program is known as Cygwin that has a complex installation process due to the large size of packages of Cygwin.

B. NS-3

Network simulator version 3 (NS-3) is an open source and discrete event network simulator, it is not an extension of network simulator version 2 (NS-2) but it is a simulator that is built from scratch and is still under development away from the development roadmap of NS-2.
NS-3 simulator [12] is written in C++ language and python, networks can be implemented in pure C++ language, and optionally some parts of simulation can be implemented in python. NS-3 simulators do not use the object oriented tool command line (OTCL) APIs that is used in NS-2 simulators.
NS-3 supports simulation and emulation as well using sockets, where hardware can connect to NS-3 simulator to emulate the network using sockets. Network simulators version 3 (NS-3) generates traces that helps in the debugging of the results that are obtained from the simulation. NS-3 simulator provides a realistic environment with a well-organized source code.

C. OMNeT++

Object modular network testbed in C++ (OMNeT++) is discrete event Simulation environment developed by Andras Varga in technical university of Budapest, it models primary communication networks but due to its generic and flexible architecture, it is used in other applications like simulation of complex IT models, modeling multi processors and queuing systems.
OMNeT++ provides component based, modular, hierarchal and extensible architecture, this simulator is filling the gap between NS-2 open source simulator and OPNET license based simulator, by providing a free software with various capability. OMNeT++ modules and components [13] are programmed using C++ language, new versions of the software are programming the modules using C++ class library that consists of the kernel of the simulation. Network description (NED) is high-level language that is used to gather single components into larger components and models. The simulation environment has a graphical network editor (GNED) that contains a NED compiler, command line (Cmdenv), graphical (Tkenv), graphical tools for the analysis of the obtained results (Plove) and documentation tool.

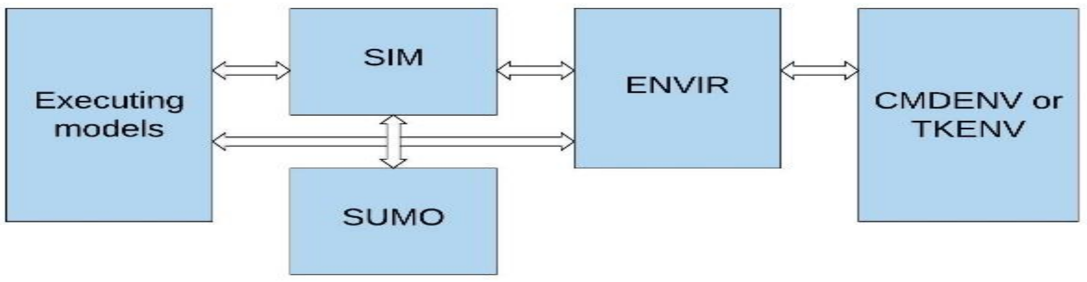


Figure 5: The architecture of OMNeT++ simulations

OMNeT++ has external extensions [13] that allows the support of wireless networks simulations. There are two important extensions are INET framework and mobility framework that is designed for mobile ad-hoc networks. INET framework and the mobility framework have good API documentation and user manual likely as any other feature in OMNeT++, which has good documentation manuals.

D. OPNET

Optimized network engineering tool (OPNET) is a licensed network simulator
that is used for simulations of both wireless and wired networks, it is licensed by Riverbed technologies. An academic edition of OPNET called OPNET IT Guru is available with free license version for researchers and teaching communities.
OPNET simulators allow user to design and study network communication devices, protocols, applications and routing protocols. OPNET simulators [14] support different wireless standards as IEEE 802.11, IEEE 802.15.1, IEEE 802.20 and satellite networks.
The main advantages of OPNET simulator is its easiness of use, friendly graphical user interface (GUI) and the good quality of documentation and technical support.

E. GloMoSim

Global mobile information system simulator (GloMoSim) is an open source simulator that is designed for mobile ad-hoc networks and its applications. It is scalable and portable simulator that has the ability to run on both shared memory and distributed memory computers [15]. It supports millions of nodes simulation and includes many sets of routing protocols. GloMoSim was retired in 2000 and then scalable commercial simulators QualNet is introduced instead of GloMoSim.
Quality networking (QualNet) simulator is very fast simulator and highly scalable that supports both wired and wireless protocols. It can simulate up to 50 thousands mobile nodes and can execute different scenarios faster than any other simulator by five to ten times. It has great graphical user interface (GUI) for custom code development. QualNet can run on Windows and Linux/Unix platforms.

4.4 Comparison between different network simulators

To summarize [16] the properties of different network simulators, Table 3 is built up to compare between these simulators.

Table 3: Comparison of studied network simulators

	NS-2	NS-3	OMNET++	OPNET	QualNet
Language supported	C++ OTCL	C++ Python	C++	C(C++)	Parsec C++
GUI Support	Poor	Good	Good	Excellent	Excellent
Time taken "learn"	Long	Moderate	Moderate	Long	Very Short
Time taken "installation"	Moderate	Long	Very short	Moderate	Short
Platform	Linux Unix Windows (Cygwin)	Linux Unix Windows	Linux Unix Windows MAC OS	Linux Windows	Linux Windows DOS
Analysis Tool	T	T	T	T	T
Trace files	T	T	T	T	T
Network Visualization Tool	T	T	T	T	T
Fast simulation	Moderate	Moderate	Moderate	Excellent	Excellent
Merits	-ease of add new protocol -large number of available protocols	-support for virtualization	-powerful GUI	-communicate with other simulators -fast -scalable	-powerful GUI -scalable -fast -emulation
De merits	-support only 2 MAC protocols. -need familiar scripting language	-python bindings do not work on Cygwin -only IPV4 support	- less number of supporting protocols -compatibility problem -slow	-commercial product -Memory consuming -insufficient tutorials	- commercial product -difficult installation in Linux

V. MIDDLEWARE FRAMEWORKS FOR VANET SIMULATORS

Middleware frameworks are required in vehicular ad-hoc networks simulators to build up the whole picture of the VANET simulator. Network simulator is to perform the simulation of the protocol or the algorithm in implemented network, and the mobility generator provides the network simulator with realistic movement traces of all motions of the vehicles of the network, to improve the realistic scenarios of the simulations. The integration between network simulator and mobility generator is done through the middleware framework that link between both simulators in order to provide the user an output that is easy to analyze and easy to go through in the related work of the user.

4.5 Federated vs integrated approach

The integration between the mobility generator and the network simulator is done through two main approaches, the federated approach and the integrated approach. The federated approach is done by building a middleware between the mobility generator and the network simulator to combine them together and to facilitate the interaction between them during the simulation. This approach has the advantage that no need to build network simulator and mobility generator from scratch, it couples between existing mobility generators and network generators. The disadvantage of this approach is that it is combining between two software of different platform and operating systems that makes the integration of them is difficult, also the two software may be combination of open source and commercial software, that makes providing fast feedback loop between both software is quite difficult.

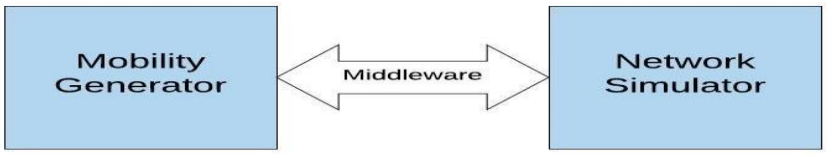


Figure 6: The architecture of a federated Mobility/Network simulator.

On the other hand, the integrated approach is done through three main ways;

a) Add communication models and networking protocols into mobility generator.

b) Add mobility models and road networks into network simulator

c) Develop new mobility and network simulator that combines between both by its nature.

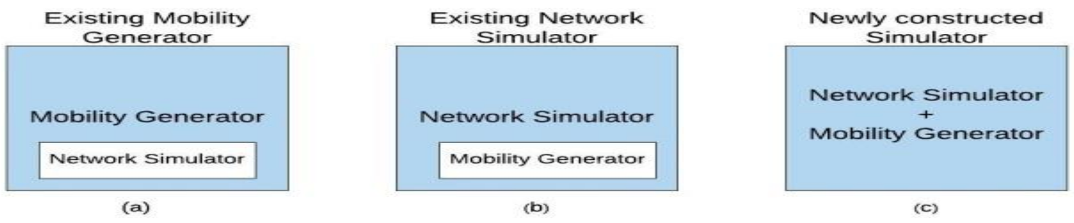


Figure 7: Three different methods of an integrated Mobility/Network simulator.

4.6 Existing middleware frameworks for VANET simulators

Many middleware frameworks integrate network and mobility simulations that are implemented in two different simulation tools, it combine both simulators in one tool with accessible graphical user interface.

In this section, different middleware frameworks are explained and hence a comparison between them is provided at the end.

A. Tra Ns

Traffic and network simulation environnement (TraNs) is a simulatio environment that integrate between both a network simulator and mobility generator in order to build a realistic vehicular ad-hoc network simulator. TraNs integrate between SUMO as mobility generator tool, and NS-2 as network simulator, it provides feedback between both vehicle behavior and mobility model.

TraNs is implemented in Java and C++ languages, and it can work under Linux and Windows platforms, it is developed in school of computer and communication sciences at EPFL, Switzerland. TraNs can work in two modes [17]: network centric mode and application centric mode. In network centric mode there is no feedback between the mobility generator SUMO and the network simulator NS-2, the mobility traces transferred from the mobility generator to the network simulator through parser with no feedback from the network simulator, the parser converts the traces from SUMO into a suitable format to be transferred to NS-2 through this parser.

In application centric mode, there is feedback between SUMO and NS-2 through an interface called TraCI,

which handles the multi dimension communication between SUMO and NS-2, hence in this mode, both mobility generator and network simulator should operate simultaneously.
TraNs is continuously developed and the new versions have several features:
(1) Realistic 802.11p support.
(2) Automatically generation of network roads from Shapefile and TIGER maps.
(3) Random vehicle routes automatic generation.
(4) Mobility trace coupling between SUMO and NS-2 through TraCI interface.
(5) Simulation of road traffic events.
TraNs is highly scalable framework, it can simulate large-scale networks (up to three thousands vehicle), and it allows for Google Earth visualization simulations.

B. GrooveNet

GrooveNet is an open source hybrid simulator that integrate between both mobility generator and network simulator, it enables communication between both real vehicles and simulated vehicles. It models inter vehicular communication (IVC) through realistic street map based topology, this facilitate the design of protocols and in-vehicle deployment. It can run simulations of thousands of vehicles in US different cities, and it can add new models for security, networking and vehicle interaction.
GrooveNet is a street map based simulator of vehicular networks that has the following features [18]:
(1) GrooveNet is an event based modular simulator that has well defined model interfaces that facilitate adding of models without any concerns related to conflicts with the existing models.
(2) GrooveNet graphical interface can automatically generate simulations across any location in the United States, consists of thousands of vehicles.
(3) GrooveNet supports three types of nodes that are used in the simulations: fixed infrastructure nodes, vehicles that are in charge of multi hopping data through one or more dedicated short range communication channels (DSRC), and gateways that are used in vehicle to vehicle communication and vehicle to infrastructure communication.
(4) GrooveNet supports global positioning system (GPS) messages that are broadcasted periodically from vehicles to inform neighbors about its current location. It supports also warning event messages and vehicle emergency messages.
(5) GrooveNet supports hybrid simulations, where real vehicles can communicate with the simulated vehicles.

C. NCTUns

National Chiao Tung University network simulator (NCTUns) is an integrated framework for vehicular ad-hoc networks that combine between network simulator and mobility generator to build up a simulator that could be used in VANET networks. Unlike TraNs, it combines network simulator and mobility generator in one module by extending network simulator to include the capabilities of mobility generators and road network simulations.
NCTUns is a C++ based simulator that runs on Linux Fedora platform. Its architecture [19] composed of graphical user interface (GUI), simulation engine (SE), car agent (CA), and signal agent (SA). NCTUns has a powerful GUI that allows the user to generate the configuration files that should be provided in the simulation of the network, this GUI include five main functions: road network construction, car profile setting, vehicle deployment, vehicle movement setting, and network topology setting.
NCTUns supports IEEE 802.11a, IEEE 802.11b, IEEE 802.11e, IEEE 802.11d, and IEEE 802.11g. It can simulate maximum number of 4096 nodes, but it has a major drawback that it runs on Linux Fedora platform, this poses great problem for many researchers and limit the usage of the tool.

D. Veins

Veins is an integrated framework that integrate between mobility generator SUMO and network simulator (OMNET++). It integrate between the two simulators via TCP socket, where the communication protocol of this socket is standardized as Traffic control interface (TraCI). This communication allows bidirectionally coupled simulation between SUMO and OMNET++, where the movement of vehicles in SUMO is directly reflected as movement of the nodes in the network simulator OMNeT++.
Veins is open source framework that is available for download free of charge, this allows researchers to use it and publish papers based on Veins, this helps in increasing the number of users. It contains large number of simulation models for vehicular networks, which are executed by the event based network simulator OMNeT++, while the mobility generator SUMO is running in parallel.

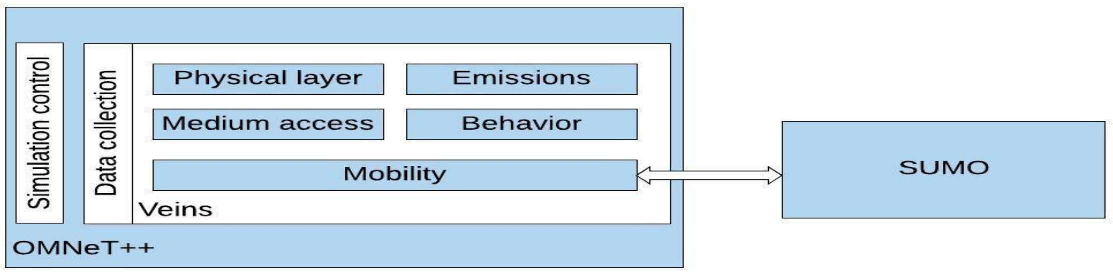


Figure 8: Veins architecture

4.7 Comparison between different middleware frameworksTo summarize [12] the properties of different middle frameworks of VANET simulators, Table 4 and Table 5 are built up to compare between these simulators.

Table 4: GUI comparison of different VANET simulators

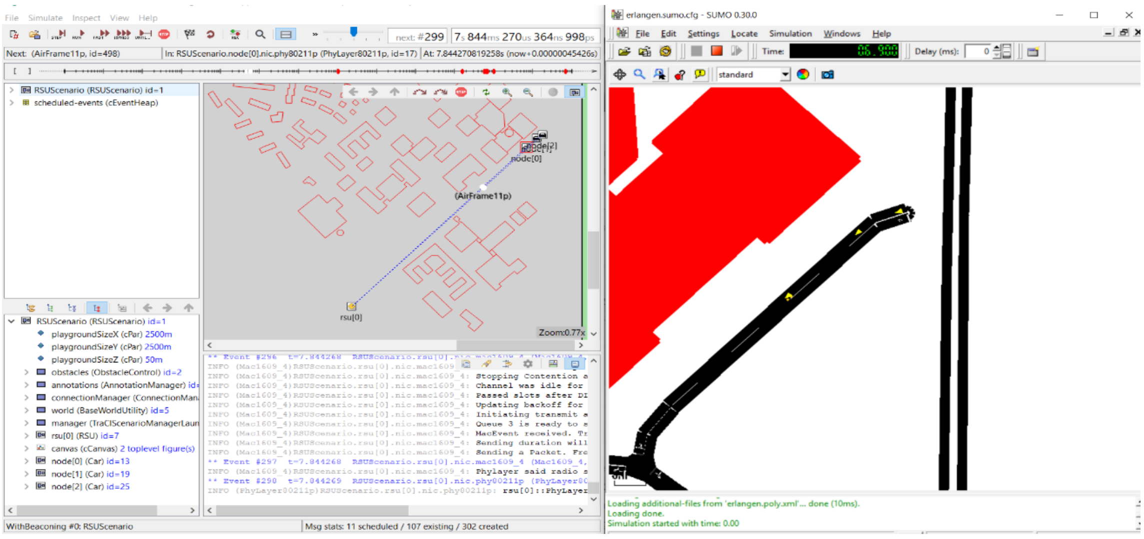
	TraNs	GrooveNet	NCTUns	Veins
User friendly	Good	Good	Moderate	Moderate
Topology view	Google earth Zoom ability No obstacles	Street view Zoom ability No obstacles	User defined No zoom ability Obstacles	Google earth Zoom ability No obstacles
Parameters input	Street map file Mobility file Graphical input	Street map file Simulation file Graphical input	Topology file Graphical input	Street map file Mobility file Graphical input
Output	NS-2 trace file .kmz file (google earth)	Simulation file Animation view	Simulation file Animation view	Simulation file Animation view

Table 5: Comparison between different middleware frameworks of VANET simulators

	TraNs	GrooveNet	NCTUns	Veins
Mobility generator	SUMO	GrooveNet	NCTUns	SUMO
Network simulator	NS-2	-	-	OMNeT++
Approach	Federated	Integrated	Integrated	Integrated
Mobility models	Random and manual routes	Random waypoint, explicit origin-destination	Random and manual routes	Random and manual routes
Simulation type	Microscopic, space-continuous and time-discrete	Microscopic, space-continuous and time-discrete	Microscopic, space-continuous and time-discrete	Microscopic, space-continuous and time-discrete
Traffic flow model	Car following and traffic assignment	Car following	Car following	Car following and traffic assignment
Lane models	Multi-lane roads with lane changing	Multi-lane roads with lane changing	Multi-lane roads with lane changing	Multi-lane roads with lane changing
Road topology	Any	Any	User defined	Any
Traffic lights	Manually defined	Manually defined	Automatically generated on intersections	Manually defined
Trip model	Random, manually defined	Djikstra, sightseeing	Manually defined	Random, manually defined
Ease of setup	Moderate	Moderate	Hard	Easy
Ease of use	Moderate	Hard	Hard	Moderate
GUI	Yes	Yes	Yes	Yes

VI. EVALUATION AND RESULTS ANALYSIS

In this chapter, Veins complete solution is used to evaluate the integration done between the mobility generator SUMO and the network simulator OMNeT++ by the framework veins. The mobility generator and the network simulator are running simultaneously, where the mobility generator simulates the real traffic simulation of the designed vehicles of the system through the mobility generator SUMO. The output of the mobility generator is trace file of the mobility patterns of the vehicles simulated in real life maps. The generated traces is then transferred to the network simulator through the traffic interface control TraCI to grantee the simultaneous operation of the mobility generator and the network simulator. The network simulator OMNeT++ receive the mobility traces as input to its simulation and translates the movement of the vehicle in real world to the movement of the nodes in the designed network. The network simulator then runs its simulation and the designer could analyze the completely designed networks and evaluate the network parameters that is targeted from the simulation of the network.


File Simulate Inspect View Help
next: #299
7s 844ms 270us 364ns 998ps
node[2]
node[0]
(AirFrame11p)
rsu[0]
Zoom:0.77x
erlangen.sumo.cfg - SUMO 0.30.0
File Edit Settings Locate Simulation Windows Help
Time:
Delay (ms):
standard

Next: (AirFrame11p, id=498) | In: RSUScenario.node[0].nic.phy80211p (PhyLayer80211p, id=17) | At: 7.844270819258s (now+0.00000045426s)

```
RSUScenario (RSUScenario) id=1
scheduled-events (cEventHeap)
```

```
RSUScenario (RSUScenario) id=1
    playgroundSizeX (cPar) 2500m
    playgroundSizeY (cPar) 2500m
    playgroundSizeZ (cPar) 50m
  > obstacles (ObstacleControl) id=2
  > annotations (AnnotationManager) id=
  > connectionManager (ConnectionMan
  > world (BaseWorldUtility) id=5
  > manager (TraCIScenarioManagerLaun
  > rsu[0] (RSU) id=7
  > canvas (cCanvas) 2 toplevel figure(s)
  > node[0] (Car) id=13
  > node[1] (Car) id=19
  > node[2] (Car) id=25
```

```
** Event #296  t=7.844268  RSUScenario.rsu[0].nic.mac1609_4 (Mac1609_4,
INFO (Mac1609_4)RSUScenario.rsu[0].nic.mac1609
INFO (Mac1609_4)RSUScenario.rsu[0].nic.mac1609_4: Stopping Contention a
INFO (Mac1609_4)RSUScenario.rsu[0].nic.mac1609_4: Channel was idle for
INFO (Mac1609_4)RSUScenario.rsu[0].nic.mac1609_4: Passed slots after DI
INFO (Mac1609_4)RSUScenario.rsu[0].nic.mac1609_4: Updating backoff for
INFO (Mac1609_4)RSUScenario.rsu[0].nic.mac1609_4: Initiating transmit a
INFO (Mac1609_4)RSUScenario.rsu[0].nic.mac1609_4: Queue 3 is ready to s
INFO (Mac1609_4)RSUScenario.rsu[0].nic.mac1609_4: MacEvent received. Tr
INFO (Mac1609_4)RSUScenario.rsu[0].nic.mac1609_4: Sending duration will
INFO (Mac1609_4)RSUScenario.rsu[0].nic.mac1609_4: Sending a Packet. Fre
** Event #297  t=7.844268  RSUScenario.rsu[0].nic.mac1609_4 (Mac1609_4,
INFO (Mac1609_4)RSUScenario.rsu[0].nic.mac1609_4: Phylayer said radio s
** Event #298  t=7.844269  RSUScenario.rsu[0].nic.phy80211p (PhyLayer80
INFO (PhyLayer80211p)RSUScenario.rsu[0].nic.phy80211p: rsu[0]::PhyLayer
```

```
Loading additional-files from 'erlangen.poly.xml' ... done (10ms).
Loading done.
Simulation started with time: 0.00
```

WithBeaconing #0: RSUScenario | Msg stats: 11 scheduled / 107 existing / 302 created

Figure 9: RSU scenario of simultaneous run of OMNET++ and SUMO

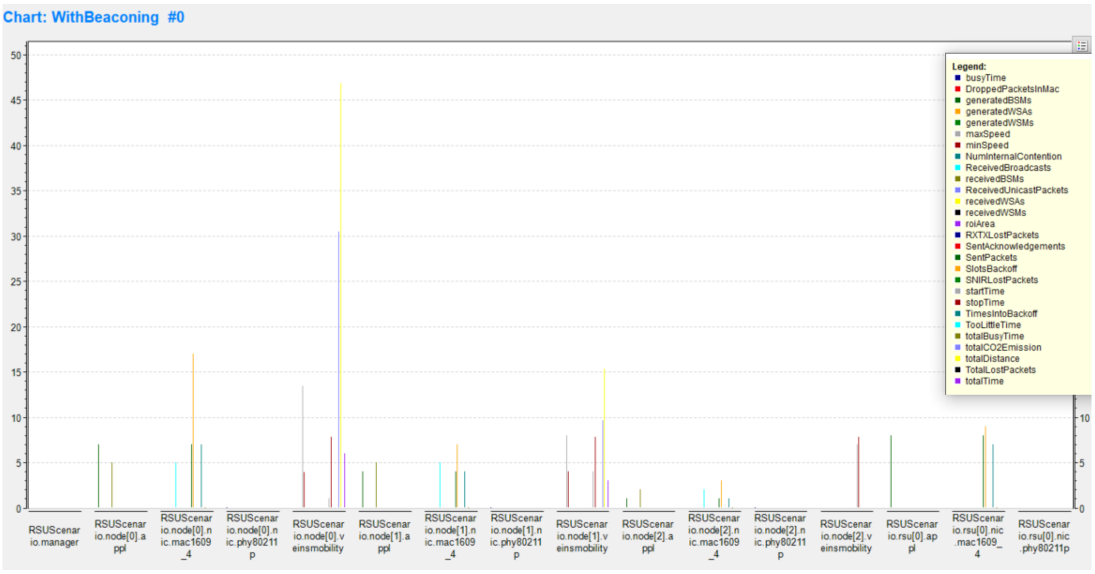


Figure 10: Results analysis of the nodes of the network

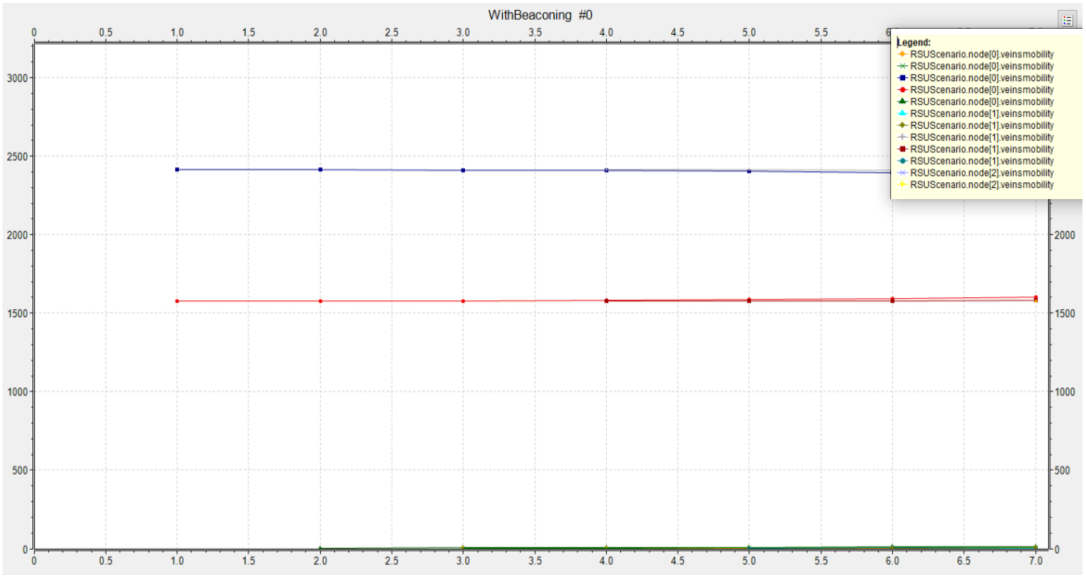


Figure 11: Mobility traces analysis of the vehicles of the network

VII. CONCLUSION

In this paper, the full architecture of vehicular ad-hoc simulators is discussed, where mobility generator is to provide the network simulator with realistic traces of the movement of the vehicles simulated in realistic environment. The traces provided from the mobility generator is to be translated in the network simulator as movement of the nodes in the networks. A middleware framework to grantee the integration between both mobility and network simulators. Each part of the three parts of VANET simulator is discussed separately followed by comparative study of the used tools per each part provided. The choice of the most suitable complete vehicular ad-hoc network VANET simulator is taken based on several factors; the application that the network is designed for, the capability of each simulation tool, and based on the financial factors that controls the usage of commercial or open source tools.
Basic simulation run is performed by integrating SUMO mobility generator with the OMNeT++ network simulator through the integrated framework Veins to build up the full design view of the network simulator, followed by analysis results of the full run scenario to show how the integration between mobility generator and network simulator is done.

REFRENCES:

1. Omkar Pattnaik and Binod Kumar Pattanayak, 2017, “Performance Analysis of MANET and VANET based on Throughput Parameter “, International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 18 (2017) pp.
2. Francisco J. Martinez, Chai Keong Toh, Juan-Carlos Cano, Carlos T. Calafate and Pietro Manzoni, 2009, “A survey and comparative study of simulators for vehicular and ad hoc networks (VANETs)”, Published online in Wiley InterScience (www.interscience.wiley.com) DOI: 10.1002/wcm.859.
3. J´erˆome H¨arri, Fethi Filali and Christian Bonnet, 2009, “Mobility Models for Vehicular Ad Hoc Networks: A Survey and Taxonomy “, IEEE Communications Surveys & Tutorials (Volume: 11 , Issue: 4 , Fourth Quarter 2009).
4. Yusor Rafid Bahar Al-Mayouf, Mahamod Ismail, Nor Fadzilah Abdullah, Salih M. Al-Qaraawi and Omar Adil Mahdi, 2016, “SURVEY ON VANET TECHNOLOGIES AND SIMULATION MODELS “, ARPN Journal of Engineering and Applied Sciences, VOL. 11, NO. 15, AUGUST 2016 ISSN 1819-6608.
5. Daniel Krajzewicz,Georg Hertkorn, Peter Wagner and Christian Rössel, 2002, “SUMO (Simulation of Urban MObility); An open-source traffic simulation “, Conference: 4th Middle East Symposium on Simulation and Modelling.
6. Vaishali D. Khairnar and Dr. S.N.Pradhan, 2010, “Mobility Models for Vehicular Ad-hoc Network Simulation “, International Journal of Computer Applications (0975 – 8887) Volume 11– No.4, December 2010.
7. Jeffrey Miller and Ellis Horowitz, 2007, “FreeSim – A Free Real-Time Freeway Traffic Simulator “, 2007 IEEE Intelligent Transportation Systems Conference.
8. J. Martinez, J.-C. Cano, C. T. Calafate and P. Manzoni, 2008, “CityMob: a mobility model pattern generator for VANETs “, ICC Workshops - 2008 IEEE International Conference on Communications Workshops.
9. Jianli Pan and Prof. Raj Jain, “A Survey of Network Simulation Tools: Current Status and Future Developments”, report.
10. Dr. L. RAJA, 2018, “STUDY OF VARIOUS NETWORK SIMULATORS “, International Research Journal of Engineering and Technology (IRJET) Volume: 05 Issue: 12 | Dec 2018
11. Teerawat Issariyakul and Ekram Hossain, “Introduction to Network Simulator 2 (NS2)”, book.
12. Atta ur Rehman Khan, Sardar M. Bilal and Mazliza Othman, 2013, “A Performance Comparison of Network Simulators for Wireless Networks “, Published in ArXiv 2013
13. András Varga and Rudolf Hornig, 2008, “AN OVERVIEW OF THE OMNeT++ SIMULATION ENVIRONMENT “, Proceeding Simutools '08 Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops Article No. 60 Marseille, France — March 03 - 07, 2008
14. Narendra Mohan Mittal and Savita Choudhary, 2014, “Comparative Study of Simulators for Vehicular Ad-hoc Networks (VANETs) “, International Journal of Emerging Technology and Advanced Engineering (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 4, Issue 4, April 2014).
15. Xiang Zeng, Rajive Bagrodia and Mario Gerla, 1998, “GloMoSim: a library for parallel simulation of large-scale wireless networks “, Newsletter ACM SIGSIM Simulation Digest Homepage Volume 28 Issue 1, July 1998
16. Arvind T, 2016, “A Comparative Study of Various Network Simulation Tools “, International Journal of Computer Science & Engineering Technology (IJCSET) Vol. 7 No. 08 Aug 2016
17. M. Piorkowski, M. Raya, A. Lezama Lugo, P. Papadimitratos, M. Grossglauser, and J.-P. Hubaux, 2008, “TraNS: Realistic Joint Traffic and Network Simulator for VANETs “, ACM SIGMOBILE mobile computing and communication review. 12.
18. R. Mangharam, D. Weller, R. Rajkumar, P. Mudalige and F. Bai, 2006, "GrooveNet: A Hybrid Simulator for Vehicle-to-Vehicle Networks," Third Annual International Conference on Mobile and Ubiquitous Systems: Networking & Services, San Jose, CA, 2006, pp. 1-8.
19. S.Y. Wang, and C.L. Chou, 2009, “NCTUns tool for wireless vehicular communication network researches “, Simulation Modelling Practice and Theory, Volume 17, Issue 7, 2009.