Modeling and Simulation of Vehicular Networks

Kaveh Shafiee¹, Jinwoo (Brian) Lee², Victor C.M. Leung¹, Garland Chow¹

¹ The University of British Columbia, Vancouver, BC, Canada
² Queensland University of Technology, Brisbane, Queensland, Australia

kshafiee@ece.ubc.ca, brian.lee@qut.edu.au, vleung@ece.ubc.ca, Garland.Chow@sauder.ubc.ca

ABSTRACT

Vehicular networks are characterized by highly dynamic network topologies, frequent network fragmentations and the fact that movements of vehicles are constrained to pre-defined roadways. Researchers have devoted considerable efforts to the development of innovative protocols and mechanisms to address the demanding quality of service requirements of various vehicular applications, taking into account of these special characteristics. Even though field testing yields more realistic results, it potentially involves more hazards and can be prohibitively expensive when done at scale. Hence, simulation has been the tool of choice for evaluating the performance of vehicular networking protocols and mechanisms. For simulating a wireless communication scenario in a vehicular networking environment, both the mobility of vehicles and the wireless communications between them should be modeled using appropriate traffic and network simulators, respectively. A conversion tool needs to be used to convert the outputs of traffic simulators to trace-files readable by network simulators. Note that in this case the generation of the trace-files takes place before the network simulation begins. However, for some vehicular applications such as safety or traffic applications, the movements of vehicles are affected by the received packets. So, both traffic and network simulators are expected to be running simultaneously and exchanging data. In this paper, we survey a comprehensive set of both traffic and network simulators as well as possible conversion tools and integration alternatives. We believe that this paper helps the researchers new to the field select appropriate vehicular network platforms and provide them with helpful insights as they run their first vehicular simulations.

Categories and Subject Descriptors
A.1 [Introductory and Survey]

General Terms
Documentation, Performance, Experimentation, Languages.

Keywords

1. INTRODUCTION

As automotive technology progresses into the 21st century, the need for wireless communications between vehicles becomes apparent. Wireless data dissemination and data delivery to/from vehicles that form a vehicular network allow a wide variety of services to support safety and non-safety applications (also referred to as comfort or infotainment applications). Safety applications include collision warning/avoidance, dynamic speed limits, traffic congestion mitigation and cooperative driving. Providing drivers with real-time safety and traffic information via wireless communications saves lives, avoids time and energy wastes and reduces pollution. Non-safety applications include Internet access, multimedia streaming, file sharing and highway toll services, to mention a few.

Communication networks which are composed of moving vehicles are characterized by the intrinsic characteristics of vehicular environments, such as dynamic topologies, high speed mobility, highly variable channel conditions and limited mobility due to roadmap constraints. As a result, readily available protocols for non-vehicular wireless networks may not meet the varying requirements of vehicular applications or may be inefficient in vehicular settings. Hence, new communication protocols need to be developed with respect to specific needs in vehicular networks and eventually deployed.

Newly designed communication protocols, network architectures and potential services for vehicular networks should be well investigated and evaluated before being deployed. Two of the most common alternatives for performance assessment are measurements over real-world test beds and computer simulations. Even though the actual testing of protocols in field experiments provides the most realistic results, it is often impractical due to cost, access and scalability constraints, particularly in the case of large networks. Field testing is potentially hazardous when the protocols are supporting safety applications such as collision warning and avoidance. Therefore, simulation has been the tool of choice among protocol and system designers. Even in case of smaller networks, computer simulations are widely used as the first step to assess newly developed protocols.

The architectures of vehicular networks are chosen primarily with consideration for the supported applications and vehicular traffic scenarios of concern, due to the diverse vehicular traffic characteristics of the highway, urban and rural street scenarios. In highways or urban areas a hybrid architecture is appropriate because vehicle-to-vehicle (V2V) communications are used in conjunction with vehicle-to-roadside (V2R) communications collaboratively in order to provide ubiquitous connectivity and better performance. In rural areas the network is expected to rely
in most situations on V2V communications for economical reasons, or cellular communications for applications that are less sensitive to costs.

In safety applications such as collision prevention or road condition notification, the delay requirements are usually very stringent. Hence, V2V data dissemination may be the only possible way to provide timely safety warnings to avoid the excessive delay of transferring data to and from the infrastructure. To balance vehicular traffic in roads, numerous traffic information systems based on both hybrid and infrastructure-less architectures have been proposed in the literature. In traffic systems with a hybrid architecture the main goal is to minimize the traveling times of individuals in the network, whereas in infrastructure-less architectures the objective is to balance the vehicular traffic in different neighborhoods. Various non-safety applications require different data communication mechanisms and architecture ranging from data dissemination in sale advertisements to data routing in Internet access, content delivery, etc. Any of these scenarios calls for a different approach in building vehicular network simulation-based test beds.

For simulating a wireless communication scenario in a vehicular environment, both the mobility of vehicles and the wireless communications between them should be modeled using appropriate simulation tools or platforms. For simulating the wireless networking protocol, a network simulator should be used, whereas the mobility of vehicles is usually simulated using a road traffic simulator. The outputs of traffic simulators, which include all vehicle positions at every time step of the simulation runtime, need to be converted to trace-files with a format readable by network simulators. For this purpose, various conversion tools have been developed. Note that in this case the generation of the trace-files takes place before the network simulation begins. However, for some vehicular applications such as safety or traffic applications, the movements of vehicles are affected by the received packets. So, both traffic and network simulators are expected to be running simultaneously and exchanging data during the simulation runtime.

Previous survey papers on vehicular network simulations either investigate the integration of only one specific traffic simulator and one specific network simulator for different scenarios and applications [1], [2], or only provide an overview of several traffic simulators for vehicular networks without considering network simulators and their live interactions with traffic simulators [3]. This motivates us to contribute this work in which we introduce a comprehensive set of both traffic and network simulators as well as possible conversion tools and integration alternatives. Our goal is to help researchers new to the field select appropriate vehicular network platforms and provide them with helpful insights as they run their first vehicular simulations. The rest of the paper is organized as follows.

Section 2 surveys the most commonly used network simulators, including their descriptions, unique features supporting the simulation of vehicular networking scenarios, and their performance comparisons. Available network simulators tend to over-simplify the propagation model of the physical channel, which might lead to unrealistic results when modeling wireless networking protocols in vehicular environments. However, the study of more precise channel propagation models for vehicular scenarios requires much further work and is out of the scope of our paper. In Section 3, we introduce a number of popular traffic simulators and explore the process and principles of generating vehicular movement scenarios in traffic simulators with an emphasis on how to generate road topologies, traffic demands and desirable outputs. Section 4 introduces a number of conversion tools as well as various integrated platforms providing real-time interactions between traffic and network simulators. A brief summary is given in Section 5.

2. NETWORK SIMULATORS

The number of existing wireless network simulators is increasing everyday making it more difficult for researchers to choose the most appropriate one that best meets their needs. Some of the most common network simulators in this area of research are Network Simulator 2 (NS-2) [4], Qualnet [5], OPNET [6], OMNeT++ [7], GloMoSim [8] and JiST/SWANS [9]. Among these network simulators, OPNET and Qualnet are not open source products and are primarily intended for commercial purposes. Due to the lack of space, in this section we briefly explain the most popular open source network simulators that are used by researchers for performance evaluation purposes in vehicular environments. NS-2 has been one of the most commonly utilized network simulators over the last decade, whereas simulators such as OMNeT++ and JiST/SWANS are gaining more popularity as they develop more advanced user-friendly features and claim to be more efficient in terms of their performances. It is worth mentioning that all these simulators are discrete event-based which means they use an event scheduler to keep track of all the events stored in an event queue and their simulation times.

2.1 Network Simulator Version 2 (NS-2)

NS-2 was originally aimed for the simulation of TCP, routing and multicast protocols over wired and wireless networks. In order to increase the processing efficiency, the major engine of NS-2 is written in C++ programming language which defines the operations of different network component objects and the event scheduler in the network. Some examples of the network objects are nodes, links, agents and protocols such as routing, transport and application protocols. Aside from C++, NS-2 also uses Tcl programming language in the form of Object-oriented Tcl (OTcl) scripts as the user interface to define the network topology as a combination of network objects and their connections and to initiate the event scheduler.

The use of OTcl scripts make the topology and scheduling changes in the network convenient and time-efficient since it removes the need for unnecessary recompilations of the C++ core as a result of any slight changes. In other words, the OTcl script controls the C++ network objects through an OTcl linkage object, which provides the connection between the C++ network object and its corresponding handle in the OTcl script, so-called OTcl handle, and also determines at what points in the simulation runtime the traffic sources should start and stop sending packets. For adding a new network object to NS-2, one should create a new network object, the corresponding OTcl handle and the OTcl linkage object that makes the connection between the C++ network object and the OTcl script possible. Due to the popularity of NS-2, many efforts have been made to adapt its modules to vehicular scenarios and to mitigate the required level of realism for vehicular networking. Some of these efforts are presented in Section 4.
2.2 OMNeT++

The building blocks in OMNeT++ simulations are called simple modules. As in NS-2, these modules which form the lowest level of simulator hierarchy are written in C++. A number of simple modules can be integrated by the user to form a compound module. Subsequently, multiple simple and/or compound modules can be linked to form a model such as a protocol. To implement the simulation scenario, the model should execute the algorithms contained in the modules. The messages that are exchanged between modules can represent frames or packets in communication networks, or jobs or customers in queuing networks. Modules can send messages directly to their destinations or alternatively to other modules along pre-defined paths. In the latter case, the exchange of messages between different modules takes place through gates, which are the input and output interfaces of modules, using inter-module connections with configurable parameters, e.g., propagation delay, bit error rate and data rate.

To define the structure of a model describing the message exchanging and communications between the corresponding modules, the user take advantage of a network description language called NED. NED plays the similar interface role as Tcl language does in NS-2. It should be noted that contrary to NS-2, which is known as a network simulator, OMNeT++ is introduced as a general purpose simulation engine in its website, which takes advantage of independently developed frameworks, also called packages, to support the simulation of communication networks. Aside from communication networks, OMNeT++ can also be used for modeling queuing networks, multiprocessors and distributed hardware systems and evaluating the performance of hardware and software architectures. Two popular frameworks that provide a comprehensive set of wireless protocols are INET and INETMANET frameworks. INET is designed for both wired and wireless networks and contains models for protocols such as IPv4, IPv6, TCP, UDP, Ethernet, MPLS, OSPF and 802.11. INETMANET is derived from INET, but specifically aimed for the implementation of routing protocols in Mobile Ad hoc Networks (MANETs), e.g., AODV, DSR. In general, both INET and INETMANET frameworks are scalable frameworks targeted to model large networks and are best suited for the simulation of network layer and higher layers in TCP/IP communication network model.

2.3 JiST/SWANS

Java in Simulation Time (JiST) is a general purpose simulation engine running atop a standard Java virtual machine. Any simulation scenario in JiST is composed of a number of entities each of them running an independent Java code in parallel. The simulation engine controls the interactions and synchronizations between different entities any time two entities need to execute some commands in their codes simultaneously. In order to simulate wireless scenarios, a Scalable Wireless Ad hoc Network Simulator (SWANS) is built over the JiST simulation engine which supports most of the protocols readily available for ad hoc networks and is also compatible with NS-2 source codes. One of the reasons JiST/SWANS is gaining more acceptance among researchers for the simulation of vehicular scenarios is the fact that the Street Random Waypoint (STRAW) [10] mobility model is specifically designed for SWANS. STRAW is an adaptation of well-known random waypoint mobility model [11] to vehicular environments. It was observed that STRAW yields more accurate simulation results compared to ordinary random waypoint model when applied to U.S. city maps [10]. Some important features of STRAW include constraining vehicle movements to street layouts defined by street maps, considering the effects of traffic congestion on vehicle mobility, supporting lane-changing on multi-lane streets and replicating simple traffic control mechanisms. It should be noted that new mobility models are included in more recent versions of STRAW [12]. The random waypoint mobility model is described in more detail in subsection 3.1.

2.4 Performance Comparison

Much research has been conducted to compare the effectiveness of different network simulators [13–16]. The most popular evaluation metrics are simulation runtime and memory consumption. The evaluation procedure is that a reference simulation scenario is defined and implemented in all of the network simulators being evaluated. The results show that JiST has a superior runtime performance that is attributed to the fact that different entities run their codes separately but in parallel. NS-2 has the slowest simulation runtime caused by the interworking between C++ and OTcl codes in its architectural design. JiST also appears to be more efficient in terms of memory consumption compared to NS-2 and OMNeT++, in which the required memory usage grows linearly as the size of the network increases. It is worth mentioning that in terms of memory consumption OMNeT++ performs slightly better than NS-2. Despite its performance shortcomings, NS-2 is the most commonly used network simulator. A review on 151 research papers in the field of wireless ad hoc network simulation in 2005 [17] showed that NS-2 had the highest usage, i.e., 43.8%, among others. This is because NS-2 has been established for a long time and since it is open source, a comprehensive set of models and protocols have been developed and contributed by researchers over this long time. As a result, beside the popular network protocols present in most other network simulators, there also exist a large number of less frequently used models and protocols contributed to NS-2. Most of these contributions are not included in the basic NS-2 installation packages, but can be found in the Contributed Code webpage [18] or in one of the numerous NS-2 forums or newsgroups. Thus, NS-2 is virtually considered as the standard for network simulation.

3. ROAD TRAFFIC SIMULATORS

Road traffic simulators also play a key role in the simulation of vehicular networks. In general, traffic simulators can be categorized as macroscopic and microscopic, depending on the level of details describing traffic flows [19]. Macroscopic models represent vehicles as an aggregated traffic flow for simplicity. On the other hand, microscopic models have the ability to describe each vehicle as an individual object with unique characteristics, and so are able to account for the complex interactions among vehicles and between drivers and transportation infrastructures.

Road traffic simulators require a complete set of inputs for reliable modeling of vehicular traffic. One of the key inputs is a road network or road topology which is represented as a set of intersections or nodes linked to each other via a set of streets or edges. Road topology data must include the information of node locations and link lengths as well as the geometric and operational details of each link, for example, number of lanes or allowable directions, curvature radius, and lane configuration at
interactions with other nearby cars, behaviors at intersections and individual drivers’ behaviors such as cars speeds, accelerations, destinations, and assigns the vehicles on the road topology moving from their origins to limits. The pause times between subsequent trips are randomly input parameters such as the simulation period, the simulation start and end times, and the simulation step size should be determined. In the subsequent subsections, we will give more details on the provisioning of any of these inputs in a number of traffic simulators.

3.1 Generic Mobility Simulation Framework (GMSF)

GMSF [20-22] developed at ETH Zurich is a microscopic traffic simulator that can employ a number of mobility models to generate realistic mobility trace-files. The road topologies originally used in the framework are derived from Swiss Geographic Information System (GIS) landscape model [23] which is not an open-source database. However, the road topologies could also be imported as networks generated by OpenStreetMaps (OSM) [24] which is a free road topology database.

The employable mobility models suggested by the developers are GIS-based mobility model [22], Multi-agent Microscopic Traffic Simulator (MMTS) model [25], Manhattan model [26] and Random Waypoint Model [11]. In the GIS-based mobility model, vehicles define steady-state random trips constrained to the road topology derived from GIS maps [23] with the possibility of keeping a safe distance from the vehicles in front of them. When moving along the trips, vehicles use the Intelligent-Driver Model (IDM) [27] to adjust their speeds and perform their car-following mechanism. Also, a simple traffic light model could be used to regulate vehicle traffic at intersections.

MMTS is a microscopic vehicular mobility model also developed at ETH Zurich. It realistically models the behaviors of all individuals living in the region of interest collected by a census. It generates mobility traces according to the trips that individuals take during their daily schedules taking their means of transportation into account for any of the three possible rural, urban or city scenarios. In the Manhattan model a grid layout is used as the road topology. Vehicles are randomly accelerating and decelerating within a permitted range of speed unless their distance to the vehicle ahead of them falls below a threshold. In this case the maximum speed in the allowable range of speed will be set to the speed of the front vehicle. The Random Waypoint Model was originally designed for ad hoc networks and may not suit the purpose of realistic vehicular mobility modeling. In this model, nodes move along a straight line from their origins to their destinations at a fixed speed in an open field with no restriction. The speed is randomly selected between a lower and an upper limit. The pause times between subsequent trips are randomly selected between zero and a maximum value. Therefore, this model cannot be used for realistic modeling of vehicle movements which are constrained to street layouts and controlled by traffic regulations.

Many formats for the outputs of GMSF are possible. GMSF can generate trace-files immediately usable by NS-2 and Qualnet network simulators. Also, the traces can be stored in a generic simulator independent XML file format which can be easily translated to any format according to users’ needs. Some realistic vehicular mobility trace-files for NS-2 which were obtained using MMTS mobility model are available in [25].

3.2 VanetMobiSim

VanetMobiSim [28-30] is a flexible vehicular traffic simulator coded in Java for modeling vehicular mobility and generating realistic vehicular movement traces for use in different network simulators such as NS-2, GloMoSim and Qualnet.

The road topology to be used in the modeling can either be defined by users or can be imported from road map databases. The user-defined topology is a graph in which intersections and streets connecting them are mapped onto vertices and edges of the graph. The imported maps could be according to Geographical Data File (GDF) [31] standard which are not freely available or else derived from maps obtained from US Census Bureau TIGER database [32] which is an open-source database.

After the topology is determined, road structure characteristics should be specified by introducing the roads with multiple lanes allowing vehicles to flow on any of the lanes in different directions and the definition of speed limits and other traffic signs on streets. Traffic flows at intersections could be regulated by stop signs or traffic lights.

For modeling the traffic demand, i.e., vehicles movement patterns, two modules have been provided in VanetMobiSim. The first module called trip generating module is responsible for defining a set of start and stop points on the road topology. The other module called path computation module is responsible for selecting the best path between the start and stop points denoted by a sequence of edges on the topology graph. The path computation logic VanetMobiSim employs is to select the path with the minimum sum of edge costs according to the Dijkstra’s algorithm. The edge costs could be determined by either their lengths, traffic congestion level, and (or) speed limits on edges.

To achieve more realistic results, VanetMobiSim has been designed to model a number of individual vehicle behaviors including smooth variations in vehicles speeds, car queues, following traffic regulations at intersections, traffic congestion and overtaking. For supporting such behaviors vehicle movements are defined as functions of nearby vehicles in their multi-lane multi-flow surroundings. For this purpose, two mobility models have been defined in VanetMobiSim. One of them is referred to as Intelligent Driver Model with Intersection Management (IDM-IM), which adapts behaviours of vehicles to the traffic signs at intersections, i.e., stop signs or traffic lights. The other one is called Intelligent Driver Model with Lane Changes (IDM-LC) which controls vehicles behaviours upon overtaking and changing lanes. It is worth mentioning that VanetMobiSim also has an incident generator feature in which one should specify the class of incident, its location and its duration.

3.3 VISSIM

In the transportation community, a number of commercial microscopic traffic simulators have been widely used as an
analysis tool for the design and assessment of a variety of transportation systems. AIMSUN, PARAMICS and VISSIM are full-featured traffic simulators providing user-friendly modeling interfaces with the ability to model large-scale transportation networks to account for area-wide impacts of localized activities or systems operation, in a sufficiently detailed level to analyze the interactions among vehicles, e.g., lane-changing or overtaking, and between vehicles and transportation systems. These traffic simulators also offer the ability to obtain detailed state variable information on each vehicle on time scales with better than second-by-second accuracy [33]. They can simulate surface street networks, freeways, interchanges, weaving and on-off ramp sections, and stop or traffic actuated controlled intersections.

For the last decade, significant advancements have been made in microscopic traffic simulation, so each software has similar strengths and capabilities nowadays. AIMSUN, PARAMICS, and VISSIM provide strong customized application development tools through Application Programming Interfaces (APIs) enabling users to develop and test advanced transportation systems and applications including, but not limited to, Intelligent Transportation Systems and wireless communications based technologies such as vehicular networks (VANETs).

This subsection describes some key capabilities of current microscopic simulation software with VISSIM as an example of this class of tools. VISSIM (“Verkehr In Städten - SIMulationsmodell”, German for “Traffic in cities - simulation model”) is the traffic simulator developed by PTV AG in Germany [35-36]. VISSIM is based on a time-step and behavior-based microsimulation model. Detailed state variable information on each vehicle can be obtained on time scale as low as 1/10 second. The basic concept of vehicle movement modeling is based on the psycho-physical driver behavior model developed by Wiedemann [37]. Speed and spacing decision of individual vehicle is determined by stochastic distributions, which have been calibrated through multiple field measurements in Germany. The traffic simulator of VISSIM allows drivers to react to not only preceding vehicles but also neighboring vehicles on multiple lane roadways as well as traffic signals in a higher alertness when approaching to signalized intersections.

VISSIM is a multimodal traffic simulator that allows users to define a full range of vehicle types including passenger cars, buses, trucks, heavy and light rail vehicles as well as pedestrians and cyclists. Traffic demands can be assigned to the network using the static or dynamic method. The static method requires users to define the traffic demand, trip start and end points, and the travel route (fixed) for the traffic. The dynamic routing method allows drivers to adaptively switch their traveling route among user-definable paths when specific events occur such as incidents or congestions. Traffic demands must be defined using origin and destination matrices with time and vehicle class dependent when using the Dynamic Traffic Assignment (DTA) feature.

For the modeling of traffic control measures such as traffic signals, roundabouts, signs, or any combination, VISSIM offers several alternatives to select depending on the operation complexity. Simple traffic signal operations (i.e., fixed-time signal control) or sign-controlled intersections can be modeled using the graphical intersection modeling interface of VISSIM. For more complex control measures such as traffic actuated control, VISSIM provides the VAP interface, a C-like traffic control macro language. VAP is supplemented with an error checking and debugging features.

One very important feature of VISSIM that sets it apart from other traffic simulators is its COM programming interface. The COM interface allows users to develop and implement their own applications on the VISSIM network using computer programming languages such as C++, Visual Basic, or Python. The COM interface provides user-developed applications with an access to the network topology, signal control, path flows, and vehicle behaviors enabling VISSIM to model complex control logic and sophisticated transportation systems [38].

### 3.4 Simulation of Urban MOBility (SUMO)

SUMO [39] is an open-source microscopic continuous-space and discrete-time vehicular traffic simulator developed at German Aerospace Centre [40]. Some important features of SUMO are the support for different vehicle types, multi-lane streets with lane changing, right-of-way rules at intersections, support of a graphical user interface and dynamic vehicular routing. Also, it accepts a large variety of network formats as inputs.

The first step in defining the simulation scenario is building the network. In SUMO the road topology can be obtained by either generating an XML file manually describing the network, importing from other software packages or using automatic topology generation functions available in SUMO. To generate the road topology by hand two files are needed; one file for defining the nodes including their IDs and positions, and another file for describing the edges between the nodes, including their start and end nodes, number of lanes, maximum speed, length, functionality and some optional parameters such as priority. Edges in terms of functionality could be considered as plain, source or sink. Vehicles are emitted to the network on source edges, whereas they are removed from the network on sink edges. Also, the assigned priorities are used in computing way-taking rules at intersections. Another possibility is to assign additional weights to edges to make them more or less attractive for vehicles to choose.

Some of the popular methods to import road topologies to SUMO include importing networks from VISUM [41], VISSIM, TIGER or OSM as well as importing ArcView [42] networks in form of ArcView databases. Recent TIGER files are stored in forms of ArcView shapefiles which are directly convertible into a network description readable by SUMO. XML files generated by OSM can be directly imported to SUMO. Alternatively, they can be imported to eWorld for editing and enrichment [43]. eWorld is a framework that can visualize and manipulate OSM mapping data before passing it to SUMO or VanetMobiSim. Some possible modifications in mapping data allowed by eWorld include the editing of names, number of lanes, speed ranges and (or) priorities, adding traffic lights and editing their logics and generating traffic events.

In addition to manual generation of topology descriptions by hand and importing them, SUMO provides the possibility of automatic topology generation including three network types of grid Networks, spider-networks and random-networks. In grid-networks the number of intersections in both horizontal and vertical directions should be determined as well as the relative distances between adjacent intersections. In spider-networks the number of axes, circles and the distance between circles are to be specified.
After the road topology has been described, traffic demand is generated. Several approaches for generating traffic demand are possible in SUMO. Some of these approaches are the definition of *trips*, *flows*, *turning probabilities*, or *routes*. Both *deterministic* and *random* routes can be defined or can be imported from other simulation tools. The parameters used in defining a *trip* for a typical vehicle are the origin or the starting edge, the destination or the ending edge and the departure time which determines when the vehicle is emitted to the network. If the trip describes periodic vehicle emissions to the network, other parameters such as period and (or) number of vehicles to be emitted are also required.

Other than using the definition of a trip for periodic vehicle emissions, one may define a *flow* to describe several vehicles using the same trip. Again, in the definition of a flow the starting and ending edges should be determined. However, instead of the departure time, begin and end times, and the number of vehicles to be emitted are determined. The time interval between begin and end times is divided by the number of vehicles to obtain departure times uniformly spread within the interval. Another possibility in defining flows is to leave out the ending street in their descriptions and include *turning probabilities* at junctions. In this case, vehicles will leave the network any time they arrive at a sink edge.

As the next step in the process of defining the simulation scenario which is optional, SUMO can employ the *dynamic user assignment* approach in [44] to avoid congestion by changing vehicle paths adaptively. Upon the completion of all these steps, the topology description and movements of vehicles are available. In the final step the corresponding files are fed to SUMO along with the simulation begin and end times to perform the simulation. SUMO can use various formats to generate its output files. One possible output format is the raw vehicle state dump which contains the positions and speeds of all vehicles in the network at all simulated time steps and is usable by NS-2 after necessary conversions.

3.5 Summary

A large number of mobility parameters and features must be enumerated and assigned to different traffic simulators. It is thus evident that selecting an appropriate traffic simulator depends on many factors. Some important factors are the required level of details, the applications and services to be offered, which may or may not be affected by different features or mobility models supported by the traffic simulator, and very importantly the need to interlink the selected traffic simulator to the selected network simulator. Among the traffic simulators reviewed, SUMO is the most popular one and as described in Section 4, SUMO is the most commonly utilized road traffic simulator for integration into and interworking with other simulators. The strengths of SUMO can be summarized as follows: it provides the flexibility of working over various operating systems. Also, its simulation execution speed is much faster than other traffic simulators, which can be attributed to its car-following model. The car-following model employed may not include as many details as the models used in some other traffic simulators do. Another reason of SUMO’s popularity is its open source license and its active open source community which improves it and keeps its features updated.

4. INTERLINKING TRAFFIC AND NETWORK SIMULATORS

The vehicle positions and traffic states which have been obtained via traffic simulators are required to be converted to trace files usable by wireless network simulators. In this regard, many *conversion tools* have been developed. Some popular examples are TraceExporter [45], MObility model generation for VEhicular networks (MOVE) [46], and TRaNSLiTe [47], which will be described in subsection 4.1. Conversion tools only take into account the effects of vehicle movements on wireless communications. In other words, static vehicle traffic traces are obtained from traffic simulators and imported to network simulators before the network simulation process begins. However, for some vehicular applications the data vehicles receive during the simulation runtime also influences drivers’ behaviours and as a result, vehicle movements. Therefore, for more realistic investigation of these applications a coupled linking between the traffic and network simulators, which supports a bi-directional exchange of information, is required. Some examples of these applications are the case in which a vehicle receives the updated traffic conditions and recalculates its path to take the path with minimum travel time, or in the case of a collision or an emergency in which the vehicle should stop or change its movement direction to avoid the collision. In this regard, various *integrated platforms* providing real-time interactions between traffic and network simulators will be introduced in subsection 4.2.

4.1 Conversion Tools

**TraceExporter**

TraceExporter coded in Java converts raw vehicle position dumps obtained as output files in SUMO to trace-files readable by NS-2. Since NS-2 only accepts positive x and y coordinates for the positions of nodes, the conversion should shift all node positions with negative coordinates accordingly. As a result of the conversion, three output files are obtained all in the form of *tcl* script files. Note that in NS-2 all information about the simulation scenario, e.g., the topology and events, should be written in *tcl* programming language in the form of *tcl* script files. One of the files is a *config* file describing the configuration of the simulation scenario. Another file is an *activity* file specifying the start and stop times of the vehicles movements. The third file is a *mobility* file describing the actual movements of vehicles during the simulation time. The traffic simulation scenario parameters should be read from the *tcl* files for use in the main *tcl* file controlling all the events in the simulation scenario by using the *source* command.

**MObility model generation for VEhicular networks (MOVE)**

MOVE is another open-source conversion tool built atop SUMO which generates mobility trace-files employable by both NS-2 and QualNet. One of its features is a set of Graphical User Interfaces (GUIs) which helps generate mobility scenarios more quickly. Also, by using these GUIs users can avoid having to write scripts from scratch for describing the traffic simulation scenario and they do not need to learn about the details of the underlying traffic simulator (SUMO).
Traffic Control Interface (TraCI)

TraNSLite is another tool for fast generation of realistic mobility trace-files for NS-2 from SUMO. It also supports map cropping and speed rescaling if the maps are imported to SUMO from TIGER and can be used anywhere a Java Runtime Environment is running.

4.2 Integrated Platforms

Two types of integrated platforms are distinguishable in vehicular network simulation community. In the first type, readily available traffic and network simulators are combined with the help of an interface tool interlinking them in a bi-directional way. In this regard the most commonly used interface tool, i.e., Traffic Control Interface (TraCI) [48, 49] and its architecture are briefly studied in subsection 4.2.1, followed by the introduction of a number of integrated platforms of the first type. In the second type of integrated platforms the traffic and network simulators are exclusively designed to interwork with each other. An example of these types of platforms is studied in subsection 4.2.2.

4.2.1 Integrated Platforms Utilizing Interface Tool

Traffic Control Interface (TraCI)

TraCI is an open-source interface that couples a road traffic simulator and a network simulator to enable real-time interactions. As it will be observed in future subsections, one of the important features of TraCI is that a large variety of traffic and network simulators and different combinations of them can be interlinked via TraCI. The main idea of TraCI is to control and access the traffic simulator via the network simulator by means of an application running over the network simulator. For this purpose, a TraCI-server component and a TraCI-client component have been developed atop the traffic and network simulators, respectively. The two components use a TCP connection for exchanging data.

TraCI works as follows. At every simulation step, the network simulator sends a simulation command to the traffic simulator including vehicle positions and the simulation times which are used for synchronization purposes. Upon the reception of this simulation command, the traffic simulator performs the next simulation step and sends back the resulting vehicle positions to the network simulator. Based on these new received positions the decisions are made and the respective commands will be sent to the traffic simulator in the next step.

So far, TraCI-Client implementations for NS-2, OMNeT++ and SWANS++ have been provided. More information regarding implementation details and latest versions of TraCI can be found in TraCI wiki [48].

Traffic and Network Simulation (TraNS)

The integrated platform that couples SUMO and NS-2 with the help of TraCI interface is referred to as the Traffic and Network Simulation (TraNS) [47, 50]. The developers of TraNS claim that TraNS could be used to couple any traffic simulator with any network simulator [50]. The implementation of TraNS for integrating SUMO and JiST/SWANS is investigated in [51].

TraNS provides the possibility to model traffic and safety events, such as traffic congestion and collisions. The events could be generated either at a specific location or via a specific vehicle. Also, a number of mobility models have been designed in TraNS to simulate individual vehicle behaviours in critical situations including Road Danger Warning and Dynamic Rerouting to provide safety and traffic efficiency. It is worth mentioning that the development of TraNS is currently suspended [47].

Vehicle in network simulation (Veins)

The other integrated platform that integrates SUMO as the traffic simulator and OMNeT++ as the network simulator through TraCI interface is Vehicle in network simulation (Veins) [52]. OMNeT++ has various frameworks each developed for a specific purpose and provides protocols relevant to that purpose. The TraCI is fully supported on both INET and INETMANET frameworks. Instructions on how to set up OMNeT++, INET framework, and the respective TraCI-client implementation is given in [52].

Multiple Simulation Inter-linking Environment for C2CC in VANETs

Generally speaking each simulator focuses on a specific set of applications and protocols of vehicular networks. Some researchers believe that the possibility of integrating various simulator environments keeps all their advantages while some disadvantages are removed. In this regard, a platform was developed in [53] to integrate NS-2 as the network simulator, VISSIM as the traffic simulator, Matlab/Simulink [54] for application development and Click [55] for realistic routing protocol simulations. Among all the integrated simulators NS-2 controls the platform in a centralized manner by communicating with the other simulators. Hence, NS-2 can directly influence the mobility of vehicles and drivers behaviors.

4.2.2 Exclusively Designed Integrated Platforms

NCTUns

Even though NCTUns [56, 57] was originally developed as a network simulator, its more recent version is an integrated platform capable of supporting the interactions between the network and traffic simulators. Some vehicle movement parameters in the traffic simulator are initial and maximum speeds and initial and maximum accelerations. The moving paths and speeds of vehicles at each point in time are determined based on the traffic conditions in their surroundings at that point. Also, an intelligent driver behaviour module is included in NCTUns which provides features such as car-following, lane-changing, overtaking, turning and obeying traffic lights. Different agents in the platform communicate by means of the TCP/UDP/IP protocol stack. The fact that the road traffic is highly integrated in the platform makes the use of external traffic simulators in the platform difficult if not impossible, which in turn makes it a less favorable option for those researchers who wish to have more freedom in selecting traffic simulators.

5. SUMMARY

In the study of vehicular communications, due to high costs of real-world experiments and scalability problems, simulation tools have gained widespread acceptance in the research community for evaluating new protocols and mechanisms in vehicular networks. The simulation of vehicular scenarios is comprised of the simulation of wireless communications in a network simulator and simulating the movements of vehicles in a traffic simulator. We have introduced some of the most popular network simulators with the focus on open source discrete event-driven simulations. A comparison of the introduced simulators indicates that JiST is
both faster and more efficient with respect to memory consumption. However, NS-2 has been the most popular network as it was developed much earlier than other network simulators and also due to the wide range of available models that have been developed for it covering all OSI layer protocols for various types of communication networks. A number of popular road traffic simulators have been introduced and their unique features and the types of scenarios and applications they are intended for were discussed. Among the traffic simulators reviewed, SUMO is most commonly used in relevant studies as it can be easily adjusted to work in various operating systems and due to its faster execution speeds. We have also surveyed several available approaches for one-way or bi-directional connection of traffic simulators to network simulators. A number of conversion tools have been introduced for importing outputs of traffic simulators to network simulators, and same integrated platforms have been reviewed for applications which require real-time interactions between traffic and network simulators.

6. ACKNOWLEDGMENTS
This work was supported in part by grants from the NSERC DIVA Strategic Research Network, the Nokia University Relations Program, AUTO21 under Canadian Network of Centers of Excellence Program, and the BC-China ICSD Program.

7. REFERENCES


