Received April 26, 2019, accepted May 26, 2019, date of publication June 13, 2019, date of current version July 17, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2922766

Vehicular Network Simulation Environment via Discrete Event System Modeling

LE WANG, RENATO IIDA, AND ALEXANDER M. WYGLINSKI, (Senior Member, IEEE)
Wireless Innovation Laboratory, Department of Electrical and Computer Engineering, Worcester Polytechnic Institute, Worcester, MA 01609, USA

Corresponding author: Le Wang (lewang@wpi.edu)

This work was supported by the MathWorks Inc.

ABSTRACT A vehicular ad-hoc network (VANET) enables vehicles to communicate with each other directly or via roadside infrastructure in order to improve road safety and efficiency. Within a VANET, communications can potentially alter vehicular mobility and conversely, that the mobility could potentially influence vehicular communications. Therefore, a VANET simulator environment is needed that can accurately model interactions between vehicular mobility and network protocols. In this paper, we present a novel vehicular network simulation environment designed using the MATLAB discrete event system (DES) in the SimEvents toolbox. The proposed simulation environment is a bit-accurate, discrete event simulator that integrates vehicular mobility operations with wireless network communication. This paper provides details on the design of the proposed simulator. Its computational costs are evaluated in terms of events quantities and execution time. The physical (PHY) layer of the proposed simulator shows a more realistic packet success rate (PSR) using bit-level processing techniques when compared with the packet-based NS-3 simulator. The performance of the priority-based media access control (MAC) layer proves the data with different priorities that can coexist in the same channel.

INDEX TERMS Vehicular network simulation, vehicular mobility models, discrete-event system.

I. INTRODUCTION

The vehicular inter-communication concept is motivated by the opportunity to improve road safety and efficiency. Several safety applications benefit by supporting direct vehicle-to-vehicle (V2V) communications [1], including accident prevention applications and lane changing applications. In vehicle-to-infrastructure (V2I) communications [2], the roadside unit (RSU) can gather and analyze traffic status information, as well as guide vehicles within an area to improve traffic efficiency [3]. At the 15th Intelligent Transportation System (ITS) World Congress [4], it was mentioned that vehicular networks have the potential to save time and save lives. Therefore, vehicular network applications can be classified into two types: *safety applications*, and *efficiency applications*.

In 2011, the U.S. National Highway Traffic Safety Administration (NHTSA) released the final report of Vehicle Safety Communications - Applications (VSC-A), which summarized eight crash scenarios based on the statistics of vehicle accidents within the U.S. with respect to frequency, cost, and casualties [5]. To address these crash scenarios, VSC-A proposed several safety applications such as Emergency Electronic Brake Lights (EEBL) [6], Blind Spot Warning (BSW)/Lane Changing Warning (LCW) [7], and Intersection Movement Assist (IMA) [8]. These safety applications are often evaluated using vehicular mobility models [9].

A vehicular network is considered to be a complicated operating environment since it must account for both vehicular mobility and communication network simultaneously. The position and speed of vehicles could potentially impact the quality of wireless communications, and the information shared over the vehicular communication network could influence vehicular path and mobility decisions. This interaction requires having the traffic mobility simulators to work closely with vehicular network simulators [10].

In general, these types of simulators have often been created and controlled separately from one another, and thus their interactions have rarely been considered. Over the past several years, vehicle networking researchers have worked on creating an interface between these two simulation

The associate editor coordinating the review of this manuscript and approving it for publication was Zhiwu Li.

environments and several approaches have been proposed [11]–[14]. Based on the level of interaction, we classify the simulators as *joint simulators* and *integrated simulators*.

For the *joint simulator* approach, an interface is created to associate the existing traffic mobility simulators with the network simulators. An example is the iTetris [11] project, which associates the traffic simulator SUMO [15] with the network simulator NS-3 [16], or OMNet++ [17] simulation environment. Another example is using TraCI [18] to connect SUMO with another simulator such as OMNet++ or MATLAB. The interface here performs the role of relaying messages between the simulators. Traffic flows are extracted from SUMO and sent to the network simulator through the interface, and conversely the instructions from the network simulators are sent to SUMO in order to alter the traffic behavior. The advantage of this cross-layer joint approach is that one is able to enjoy the benefits of both well-developed simulators. However, one limitation of this approach is the design complexity of the interface since it needs to let both simulators operate simultaneously. Another limitation of this approach is the configuration complexity, since the users often need to tweak a large number of parameters for both simulators in order to make the overall simulation experiment work correctly.

The alternative approach to combine the network and traffic simulators is to use them into one single simulator in order to achieve full interaction. This type of simulator is called an *integrated simulator*, which has the capability having both simulators directly work and interact with each other. Several examples include MoVes [12], NCTUns simulator [14], and VISSIM [13]. The limitations for this approach mainly come from an over-simplified communication network. For example, several simulators only have a basic radio propagation model with Carrier-sense multiple access with collision avoidance (CSMA/CA) as the MAC layer [19].

In this paper, we present an integrated vehicular network simulation environment, which we refer to as *VANET Toolbox*, that functions in the MATLAB/Simulink environment as shown in Figure 1. The proposed simulator consists of a Simulink library with custom-built blocks covering the main stack of vehicular network protocols including the application (APP) layer, the medium access control (MAC) layer, and the physical (PHY) layer as shown in the figure. Several mobility operations including car following model, lane changing model, as well as intersection management that are embedded in the APP layer. The design objective of the proposed simulator is to provide a vehicular simulation environment to enable research and development in this expanding field. Table 1 provides a summary of the acronyms used in this paper.

The contributions of this paper include the followings:

- The proposed novel vehicular network simulation environment is an integrated type simulator combining both vehicle traffic simulation and network simulation together. It supports a hybrid of time-driven and event-driven simulation environment.
- A PHY layer that precisely models bit-level processing techniques, which is essential to emulate precise and realistic wireless channels.
- Performance evaluation of vehicular networks across the PHY layer and the MAC layer. The results proves the effectiveness of the proposed simulation environment.

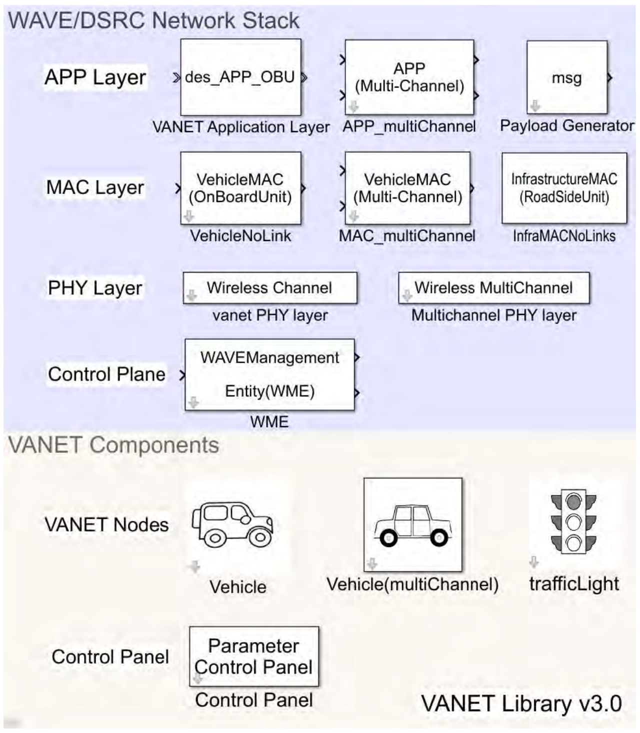


FIGURE 1. VANET Toolbox consists of a Simulink library containing vehicular network blocks. One can create simulation models by dragging the needed blocks from the library to an empty Simulink model.

The rest of paper is organized as follows: Section II provides an overview to the Discrete Event System (DES) theory and MATLAB DES framework. Section III describes the design architecture of the proposed simulation environment including PHY, MAC and APP layers. Section IV shows several simulation scenarios with different mobility models. Testing results and performance evaluations of the proposed simulator are shown in Section V. Section VI concludes the paper.

II. OVERVIEW TO DISCRETE EVENT SYSTEM

A. DISCRETE EVENT SYSTEM

A *system* is a set of interacting components which behave together to perform a function and this function cannot be performed by any of the individual parts [20]. At a specific time, a system's behavior can be described in a measurable way, *i.e., state*. The state of a system at time t_0 is defined as the output y(t) of a system for all $t >= t_0$ is uniquely determined by the system status at t_0 and system input $u(t)$, $t >= t_0$. Given the initial condition $\vec{x}(t_0) = x_0$ and the input $\vec{u}(t)$ for all $t >= t_0$, the state $\vec{x}(t)$ is presented by *state equations* shown

TABLE 1. Acronyms used in this paper arranged in alphabetical order (A-Z).

Acronyms	Meaning	Acronyms	Meaning
AC	Access Category	L-SIG	Legacy Signal field
ACK	Acknowledgment	L-STF	Legacy Short Training Field
AIFS	Arbitration Inter-frame Spacing	MAC	Media access control layer
AWGN	Additive White Gaussian Noise	Non-HT	Non-high-throughput
BPSK	Binary Phase Shift Keying	OOP	Object oriented programming
BSM	Basic Safety Message	OS	Operating system
BSW	Blind Spot Warning	PDR	Packet Delivery Rate
CFM	Car following model	PER	Packet-error rate
CRC	Cyclic Redundancy Check	PHY	Physical layer
CT	Channel tracking	PLCP	Physical Layer Convergence Protocol
CW	Contention window	PSDU	PLCP Service Data Unit
DCF	Distributed Coordination Function	PSR	Packet success rate
DES	Discrete event system	QoS	Quality-of-Service
DIFS	DCF Inter-frame Space	RDT	Reliable Data Transmission
DSRC	Dedicated Short Range Communication	RSU	Road side unit
EDCA	Enhanced Distributed Channel Access	Rx	Receiver
FIFO	First-in, first-out	SAE	Society of Automotive Engineering
HCF	Hybrid Coordination Function	SIFS	Shortest Inter-frame Space
ISI	Inter-symbol interference	SN	Sequence number
ITS	Intelligent Transportation System	Tx	Transmitter
LCM	Lane Changing Models	V2I	Vehicle to infrastructure
L-LTF	Legacy Long Training Field	V2V	Vehicle to vehicle
LOS	Line-of-sight	VANET	Vehicular ad-hoc network

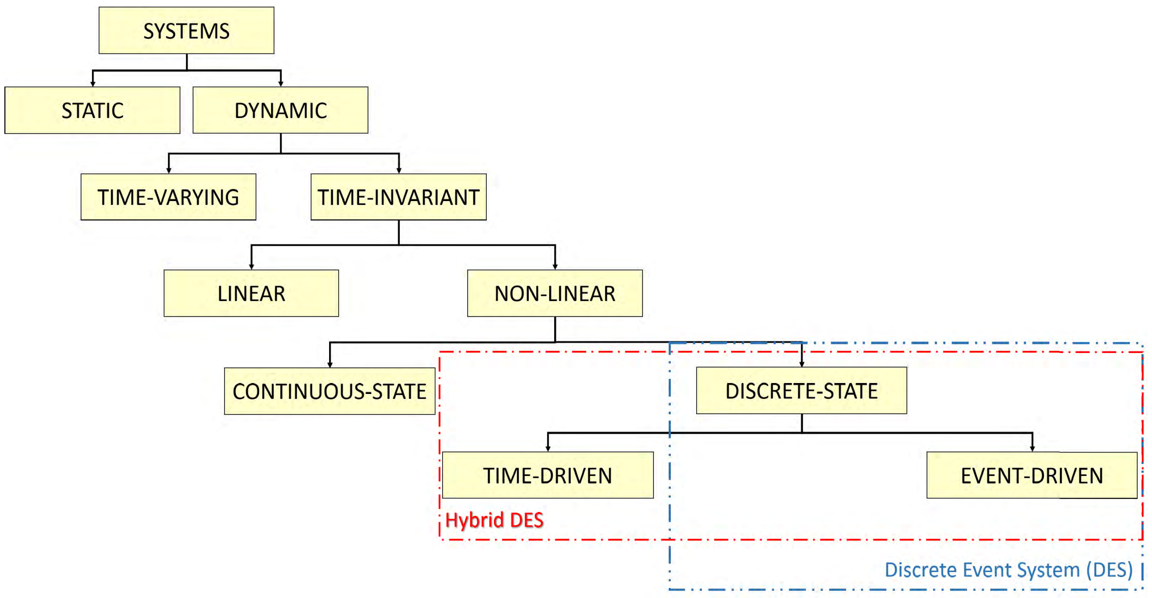


FIGURE 2. Systems overview and Discrete Event Systems. Discrete Event System is classified as a discrete-state event-trigger system. A hybrid system including both time-driven and event-driven DES is suitable for network PHY layer simulation.

in Eq. (1):

$$\dot{\vec{x}}(t) = f(\vec{x}(t), \vec{u}(t), t), \quad (1)$$

Then the output $\vec{y}(t)$ is determined by *state equations*, *input*, and *time* shown by:

$$\vec{y}(t) = \vec{g}(\vec{x}(t), \vec{u}(t), t), \quad (2)$$

The state space of a system is a set of all possible values a state may take. Based on the type of states in a model, a system can be classified into either a continuous-state system or a discrete-state system, as shown in Figure 2 [21]. In a continuous-state system, the time variable, t, enables the system to transit from one state to another state continuously. A system with such property is referred to as *time-driven system*. A continuous-state system is considered to be time-driven.

In a discrete-state system, the states are only allowed to change from one discrete state value to another. The state transitions are either synchronized by a clock or occurred asynchronously at some special time point due to events. An *event* can occur instantaneously and cause state transitions.

If state transitions in a discrete-state system are synchronized by clock ticks, it is a *discrete-state, time-driven* system. Otherwise, if state transitions occur asynchronously at various random time instants due to events, this system is referred as a *discrete-state, event-driven* system or, shortly, a discrete-event system (DES) as shown in Figure 2 . The set of events serves the purpose of driving a DES since each event may cause a state transition.

Even though a DES is defined as a discrete-state event-driven system, it may be modeled as event-driven and/or time-driven. In fact, a hybrid DES when both time-driven and event-driven are present is more general, as shown in Figure 2. For example, the operating system (OS) in a computer is designed to not only respond to asynchronous events that have occurred at any time but also process functions synchronized by the computer clock.

A continuous state system can be modeled either by differential equations (continuous time) or difference equations (discrete time), while a discrete-state event driven can be analyzed by *automata*. For instance, suppose we have an event set $E = \{a, b, g\}$. The *state space* of the automation is $X = \{x, y, z\}$. The transition function of the automation is denoted as $f : X \times E \to X$. For example, $f(x, g) = y$ means that the automation is in state x, after event g happened, the automation transits the state to y, *i.e.,* $x \to y$. The trigger event g may be either an external input to the system, or an event generated by the system itself. The initial state is denoted by x_0. A deterministic automation, denoted by G, is defined in Eq. (3) [21]:

$$G = (X, E, f, \Gamma, x_0), \tag{3}$$

where Γ is the active event set of G at x. Each state has a feasible events set $\Gamma(x)$, the events from $\Gamma(x)$ are the only events which may occur at this state.

B. DISCRETE EVENT SIMULATION

In order to perform DES simulation, the events sequences need to be associated with clocks. Suppose we have a DES with a single event $E = \{\alpha\}$. The feasible event set $\Gamma(x) = \{\alpha\}$ for all $x \in X$. The event sequence in this DES is $\vec{e} = \{e_1, e_2, \ldots, e_k\}$ and $e_1 = e_2 = \ldots = e_k = \alpha$. *Event Lifetime* denoted by v_k is defined as the length of the time interval of the two successive events. For the single event DES, the kth lifetime of the event is defined as:

$$v_k = t_k - t_{k-1}, \quad k = 1, 2, \ldots, k, \ v_k \in \mathbb{R}^+, \tag{4}$$

At time t_{k-1}, the kth event, e_k, is *enabled* with a lifetime v_k. A timer attached to e_k starts to count down from v_k. At time $t_k = t_{k-1} + v_k$, the timer reaches 0, e_k has to occur, a state transition is caused from x_{k-1} to x_k. Then, the same process repeats with the $(k+1)$th event, e_{k+1}. Thus, a DES can be specified by the *clock sequence* of events, that is, $\vec{v} = \{v_1, v_2, \ldots, v_k\}$.

A state x, including the initial state x_0, has clock values y_i where $i \in \Gamma(x)$. The *triggering event* e' is the next event which will occur at that state x, *i.e.,* the event chosen with the smallest clock value defined by:

$$e' = \arg\min_{i \in \Gamma(x)} \{y_i\}, \tag{5}$$

When event e' occurs at state x, a new state x' is generated from the state transition function $f(x, e')$. The *inter-event time*, y^*, represents the amount of time spent at state x and is calculated by:

$$y^* = \min_{i \in \Gamma(x)} \{y_i\}, \tag{6}$$

The updated time is obtained by $t' = t + y*$. Once the new state x' is generated, the clock values for all feasible events are updated. If an event $i \in \Gamma(x')$ and $i \neq e'$ remains feasible in the new state x', the new clock value is $y_i' = y_i - y*$. For all events which are not feasible in x but become feasible in x', *i.e.,* $e' \in \Gamma(x')$ but $e' \notin \Gamma(x)$, a set of new lifetimes are supplied by the DES.

From a computer implementation standpoint, a DES simulator should have an event scheduling scheme, *i.e.,* a variation of timed state automaton DES model, such that whenever an event i is enabled at time t_n, its next occurrence is scheduled at time t_n+v_i, where v_i is a lifetime sample supplied by the DES. Thus, a Scheduled Event List (SEL) replaces maintaining the clock values y_i, $i \in \Gamma(x)$ defined by:

$$L = \{(e_k, t_k)\}, \quad k = 1, 2, \ldots, m_L, \tag{7}$$

where m is the number of events in the events set E, m_L is the number of *feasible* events for the current state, *i.e.,* $m = |E|, m_L = |\Gamma(x)|, m_L \leq m$. The SEL is always ordered on a *smallest-scheduled-time-first* basis. Additional research efforts on DES simulation concepts can be found in references [21]–[26].

C. MATLAB DES SIMULATION FRAMEWORK

In order to simulate a specific DES of interest, there are two options: either build a DES simulator or use an existing DES simulator. Based on the event scheduling scheme and/or process-oriented scheme, it is possible to create a DES model using standard computer languages such as C++. However, building a DES simulator is outside of the scope of this paper. As our goal is to simulate vehicular network behaviors, choosing an existing DES simulator is preferred.

In this paper, the vehicular network simulator is developed using *SimEvents*, which is a MATLAB toolbox from The MathWorks, Inc. SimEvents needs to work with Simulink in a time-driven simulation environment. The MATLAB DES system object inside the SimEvents library allows users to create an event-driven system using MATLAB object oriented programming (OOP) languages. Thus, a system developed by SimEvents with MATLAB DES supports both time-driven and event-driven components via a hybrid approach. The DES model created using SimEvents is a timed DES model with a event scheduler, which maintains a scheduled event list (SEL) as we have discussed in the previous section. A DES model might contain more than one DES object. A DES object can customize the event type and its life cycle by overloading the methods, *i.e.,* member functions, inherited from the base class *matlab.DiscreteEventSystem*.

A MATLAB DES has three elements: *Entity*, *Event* and *Action*.

1) ENTITY

The elements flow in a DES are called *entities*. The entities contain static information, which can be MATLAB built-in data type or structured/bus data types. A MATLAB DES may have one or more entity storages, with each storage containing entities in certain order, such as a first-in, first-out (FIFO) queue or a priority queue. A MATLAB DES can take entities as input or output and entities can leave a MATLAB DES and enter another MATLAB DES.

2) EVENT

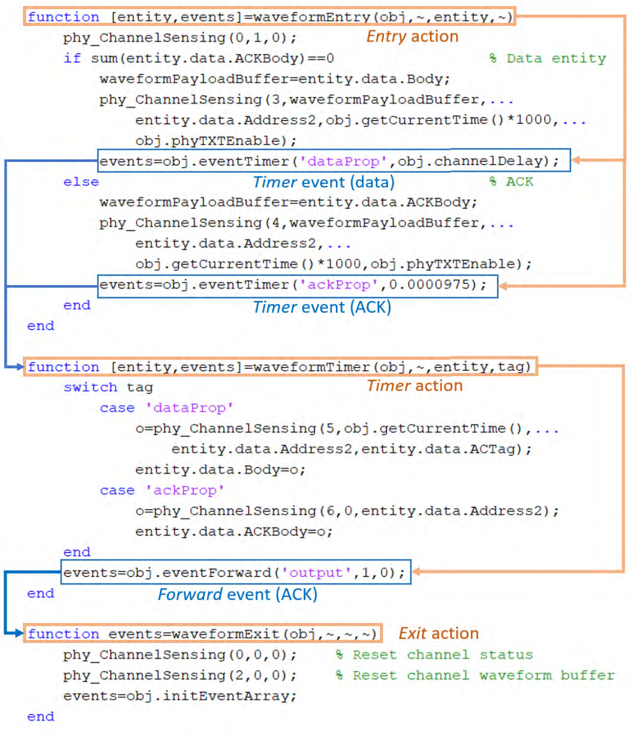
Multiple types of events can be scheduled and executed to an entity. These events model activities such as entity creation, destroy, forward (send/receive), delay and search. As of MATLAB/Simulink R2018a, MATLAB DES supports five types of events:

- Generate: *obj.eventGenerate( )* can generate an entity in the target storage.
- Destroy: *obj.eventDestroy( )* can destroy an entity in the target storage.
- Timer: *obj.eventTimer( )* delays an entity to a period of time.
- Iterate: *obj.eventIterate( )* iterates entities in the target storage with conditions.
- Forward:*obj.eventForward( )* forwards entities to a storage or an output port.

3) ACTION

When an event is due for execution, actions are invoked. These actions are conducted by user-defined methods, which may contain the algorithms. The flexibility characteristics of actions make the developers to create varieties of DES modules.

Partial code of the PHY layer implementation is shown in Figure 3. In a communication environment, the transmitter sends out a waveform, which flows through the wireless channel link and arrives at the receiver. The wireless channel link can be modeled as a discrete-event system. In the figure, the *Entry* action is triggered when a waveform entity enters the DES and stays inside the storage. In the *Entry* action method, a series of channel sensing related actions are



```
function [entity,events]=waveformEntry(obj,~,entity,~)
    phy_ChannelSensing(0,1,0);
    if sum(entity.data.ACKBody)==0                      % Data entity
        waveformPayloadBuffer=entity.data.Body;
        phy_ChannelSensing(3,waveformPayloadBuffer,...
            entity.data.Address2,obj.getCurrentTime()*1000,...
            obj.phyTXTEnable);
        events=obj.eventTimer('dataProp',obj.channelDelay);
    else                                                % ACK
        waveformPayloadBuffer=entity.data.ACKBody;
        phy_ChannelSensing(4,waveformPayloadBuffer,...
            entity.data.Address2,...
            obj.getCurrentTime()*1000,obj.phyTXTEnable);
        events=obj.eventTimer('ackProp',0.0000975);
    end
end

function [entity,events]=waveformTimer(obj,~,entity,tag)
    switch tag
        case 'dataProp'
            o=phy_ChannelSensing(5,obj.getCurrentTime(),...
                entity.data.Address2,entity.data.ACTag);
            entity.data.Body=o;
        case 'ackProp'
            o=phy_ChannelSensing(6,0,entity.data.Address2);
            entity.data.ACKBody=o;
    end
    events=obj.eventForward('output',1,0);
end

function events=waveformExit(obj,~,~,~)
    phy_ChannelSensing(0,0,0);    % Reset channel status
    phy_ChannelSensing(2,0,0);    % Reset channel waveform buffer
    events=obj.initEventArray;
end
```

FIGURE 3. Example code on the implementation of the PHY links. The PHY link DES receives a waveform entity, delays it for a predefined period, and forward out to the receiver. The operations are achieved by alternately activated events and actions.

performed and, according to the type of the waveform entity, *i.e.*, data or Acknowledgment (ACK), two separate *Timer* events are created to delay the entity for a period. Once the delay is done, the *Timer* action is activated, in which the waveform collision is simulated and the waveform entity is forwarded out of the DES via *Forward* event. After the waveform entity left the DES, the *Exit* action is triggered to reset the channel status. More details about the implementation of the PHY layer are provided in Section III.

A MATLAB DES environment is suitable for simulating network behaviors since the data flowing through different network layers can be treated as *entities*. For example, the data units of the PHY layer waveforms can be modeled as *waveform entities*. The data generation, movement, and destroy functions within a network can be implemented by *events*. Users can further define more dynamic behaviors, such as media access for the MAC layer and channel model for the PHY layer, using *actions*. In the next section, the implementation of each component in the proposed simulation environment is presented with details.

III. PROPOSED DES V2x SIMULATOR: VANET TOOLBOX

Generally, a vehicular network simulation is a combination of a time-driven system and an event-driven system. Figure 4 illustrates the design structure of two vehicular nodes communicating over a wireless channel. The framework consists

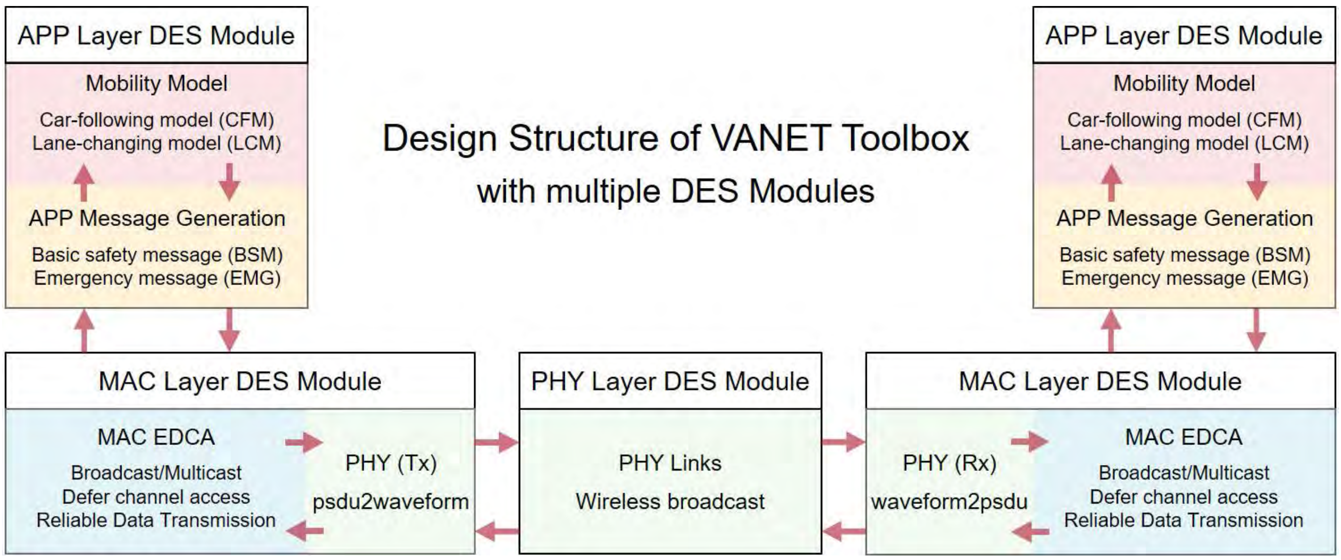


FIGURE 4. Design Structure of *VANET Toolbox*. The APP layer integrating network model and vehicle mobility model is a hybrid of event-driven and time driven. The MAC layer focuses on EDCA and is purely event-driven. The wireless channel link in PHY layer is a time-driven DES module.

of three DES modules: APP Layer DES Module (APP DES), MAC Layer DES Module (MAC DES) and PHY Layer DES Module (PHY Link DES), among which APP DES integrates vehicle mobility models with APP message generation operations, MAC DES includes MAC layer activities and PHY transmitter (Tx) / receiver (Rx) on bit-level processing and PHY Link DES only simulates wireless propagation channels.

The mobility models are integrated in the APP DES, which makes our proposed simulator an *integrated* type simulator. This design facilitates the information exchange between the vehicle mobility activities and the network communication operations. The movements of vehicles are controlled by varieties of mobility models such as car-following model (CFM) and lane-changing model (LCM). The mobility models are implemented by different safety-related or non-safety applications. According to the vehicular traffic scenario, the applications may generate messages and share with other vehicle nodes. This situation is event-driven pattern. Additionally, the applications may create beacon messages such as Basic safety messages (BSMs) at 10 Hz, which is time-driven pattern. Thus the APP DES is a hybrid of event driven and time driven. These generated messages are disseminated via wireless network communication and reciprocally the performance of network communication could affect the vehicle operations. The section only focuses on the design of network communication simulation environment, the discuss of the mobility models is presented in Section IV.

The MAC layer of a vehicular network is different compared to other WLAN devices since it grants priorities to various messages such that the messages with higher priority have shorter deference in channel contentions. This mechanism is referred to as Enhanced Distributed Channel Access (EDCA) and it is defined in the IEEE 802.11 standard [27]. Furthermore, the MAC layer is responsible for generating frames, waiting for ACKs, and initiated retransmissions when timeouts occur. All of these MAC layer behaviors are event-driven. Additionally, the transmitter (Tx) and receiver (Rx) of the PHY layer are integrated with the MAC Layer DES Module. The PHY Tx is responsible for converting the binary message information into wireless waveform symbols, while the PHY Rx is used to reverse the process. Both operations from the PHY Tx and Rx are based on bit-level processing, *i.e.,* users can manipulated every single bit of then data when necessary. The integration design of the MAC DES has two purposes. First, the PHY operations of the bit-level processing is time-continuous instead of event-driven, thus the PHY Tx/Rx cannot be implemented using DES. In our proposed simulator, a series of functions are created to perform the bit-level processing operations. The second reason is to constrain the total number of DES units in the simulation model in order to enable simulation efficiency. The creation of a DES involves overhead computational costs, including assigning input/output ports and allocating queue memories. This overhead may potentially lower the simulation speed. Thus, in our design process one of the most basic requirements is to use as few DES units as possible.

The PHY link DES module only simulates the wireless channel links since both PHY Tx and Rx are integrated with the MAC DES module. The PHY link is a relatively straightforward DES, as it is only responsible for accepting the incoming waveforms from the PHY transmitters and forwarding them to the PHY receivers after an air propagation delay. This process is an event-driven pattern. During the air propagation delay, channel models such as AWGN and two-ray ground reflection model can be applied to the waveforms and this process is implemented by functions instead of DES.

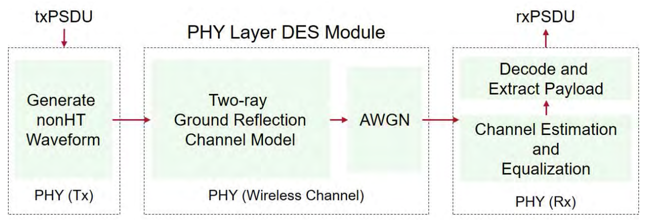

txPSDU
PHY Layer DES Module
rxPSDU
Generate nonHT Waveform
PHY (Tx)
Two-ray Ground Reflection Channel Model
AWGN
PHY (Wireless Channel)
Decode and Extract Payload
Channel Estimation and Equalization
PHY (Rx)

FIGURE 5. VANET PHY Link Modeling. The PHY Tx and Rx are for the data encoding and decoding on bit-level processing. Wireless channel link is a DES module with two-ray ground reflection model and AWGN as default.

A. PHY LAYER IMPLEMENTATION

In vehicular network simulations, a precise representation of the PHY layer is necessary in order to obtain reliable results for comparison with real hardware performance. Popular vehicular network simulation tools including VEINS [28] and iTETRIS [29], [30], which usually have a simplified PHY layer. The network simulators they adopted, NS-3 and/or OMNet++, employ abstracted PHY layer [31], where the smallest indivisible data unit used is the packet, *i.e.,* the packet is either received entirely or not at all. Several details of the wireless communication implementation, such as channel estimation, frequency offset estimation and correction, waveform modulation and demodulation, are omitted due to this abstraction. However, individual bits inside a waveform are necessary to perform accurate simulations of the PHY layer and channel models. In this section, we will introduce the proposed PHY layer with bit-level processing techniques. The performance of the PHY layer in terms of packet success rate (PSR) is evaluated in Section V.

1) DESIGN THE PHY LAYER ON BIT LEVEL

Figure 5 illustrates a basic wireless PHY link model that converts the received frame from the MAC layer into a wireless waveform and lets the waveform pass through the wireless channel. The interaction between the PHY layer and the MAC layer is handled by the Physical Layer Convergence Protocol (PLCP). A PLCP Service Data Unit (PSDU) is generated by serializing the MAC layer frame into a binary bit stream. The PSDU bits along with the PLCP preamble and header according to the IEEE 802.11 [27] are grouped into symbols and finally becomes a waveform. The wireless channel consists of a two-ray Ground Reflection Channel model and an Additive White Gaussian Noise (AWGN) model by default. This design is based on the research of line-of-sight (LOS) conditions specified in [32]. Additional channel models including *Rural_LOS* and *Urban_NLOS* can be selected during the simulation. The received waveform is decoded and verified using the bit-level receiver design in [33]. Only the PHY channel link is designed in DES, both PHY Tx and Rx are implemented using functions from WLAN Toolbox and are integrated with the MAC DES module.

Figure 6(a) illustrates the process of generating a waveform at the transmitter (Tx) at the bit level. The IEEE 802.11p PHY layer is derived from the IEEE 802.11a Non-HT

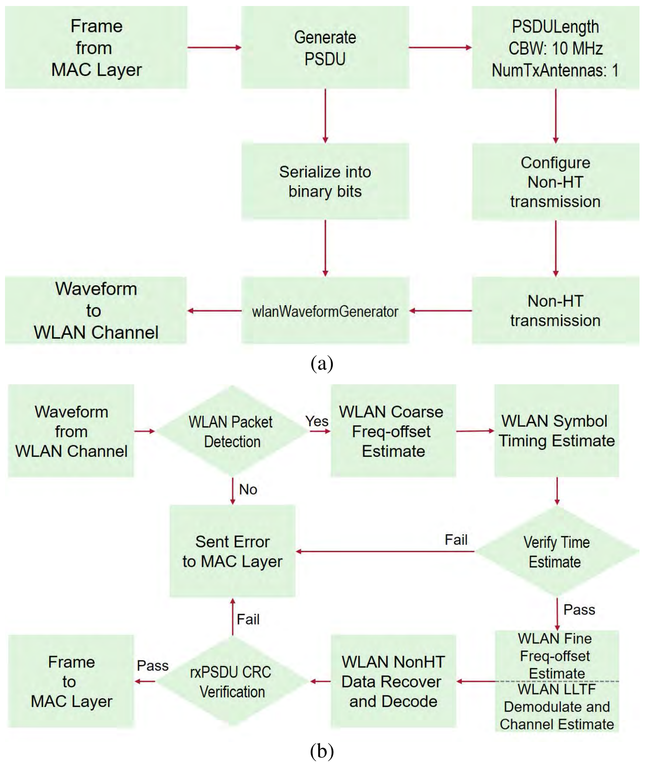


FIGURE 6. Modeling the Tx and Rx of the PHY layer on bit level. The data unit flows in the channel is wireless symbols exactly the same as the real radio transmission. (a) Bit-level processing on transmitting a waveform. The frame is converted into bits and based on the configuration on Non-HT transmission, the bits are converted into symbols and sent to the wireless channel. (b) Bit-level processing on receiving a waveform. The waveform goes through packet detection, frequency offset detection and correction, channel estimation, decoding, CRC check and finally being converted to a frame to the MAC layer.

transmission specifications. In MATLAB, *wlanNonHTConfig* creates a Non-HT object in order to configure the transmission parameters. It is configured for a 10 *MHz* channel bandwidth with a single transmit antenna according to the IEEE 802.11p standard [27]. A Non-HT Orthogonal Frequency-Division Multiplexing (OFDM) symbol consists of 64 sub-carriers with a 10 *MHz* bandwidth and symbol period of 6.4μs. A 1.6μs guard interval (GI) is inserted between each symbols in order to prevent inter-symbol interference (ISI). A PLCP header, including information of data rate and PSDU length, is prepended to the PSDU. From the perspective of a waveform, the PLCP header is the Legacy Signal (L-SIG) field and the PSDU along with a tail and padding becomes the data field. Ahead of the L-SIG field, a PLCP amble is attached, which includes a Legacy Short Training Field (L-STF) and a Legacy Long Training Field (L-LTF). L-STF is used for the packet detection, initial frequency offset estimation, and coarse timing synchronization. The L-LTF is used for the fine time synchronization, channel estimation, and fine frequency offset estimation. Thus, a complete waveform consists of a L-STF, a L-LTF, a L-SIG, as well as a data field. These fields are generated separately and concatenated to form a complete Non-HT transmit waveform. The Non-HT configuration object

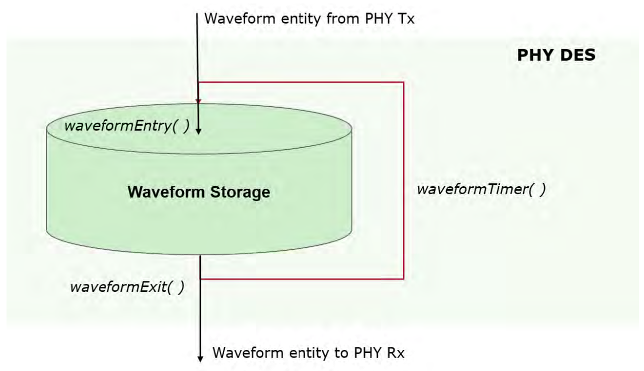


FIGURE 7. Modeling the PHY link using MATLAB DES. The waveform entity enters the DES module, stays for a period and then left the module. Only one waveform type storage is created. waveformEntry(), waveformTimer() and waveformExist() are actions.

specifies the parameters for generating the data fields of a waveform. The *cfgnonHT.PSDULength* property indicates the length of bytes to be sent in the Non-HT data field. A Non-HT waveform is then generated by function *wlanWaveformGenerator* according to the configuration of *wlanNonHTConfig*.

Figure 6(b) shows the process of payload extraction when a waveform arrives at the receiver (Rx). The first field that needs to be processed is the L-STF. In the vehicular network, L-STF has a length of $16\mu s$ with 10 repetitions. Due to its correlation properties, the first seven repetitions are used for time synchronization purposes by performing self-correlation calculations. The rest of the sequence is used for packet detection, coarse frequency offset (CFO) detection, and correction and setting the automatic gain control (AGC). The second field that needs to be examined is the L-LTF, which is composed of a cyclic prefix (CP) equaling to the period of two GIs, *i.e.,* $3.2\mu s$, followed by two identical long training symbols, *i.e,* $2 \times 6.4\mu s$. Channel estimation, fine frequency offset estimation, and fine symbol offset estimation all rely on the L-LTF. With all estimation and correction stages executed, the L-LTF demodulator and channel estimator operations are performed based on the demodulated L-LTF. Note that the demodulated L-LTF is also used for noise power estimation. Finally, the Non-HT data field is extracted and recovered into the PSDU. The integrity of received PSDU, *rxPSDU*, is verified by the Cyclic Redundancy Check (CRC). Consequently, the rxPSDU is sent to the MAC layer if it passes the CRC. The above operations of both PHY Tx and Rx are based on the bit-level processing features. This is exactly the same process when an actual waveform is transmitted among radio hardwares. Thus the PHY layer in our proposed simulator is more realistic and accurate.

2) MODELING THE PHY LINK IN MATLAB DES

The discrete-event implementation of the PHY layer link using MATLAB DES is shown in Figure 7. In the PHY DES module, only one storage resource is created to contain the *waveform type* entities. When a waveform entity enters the DES module, it stays in storage. An action called *waveformEntry()* is activated due to this 'entering' activity. In the body code of the *waveformEntry()*, a timer event is associated with the waveform entity creating a delay for a predefined period. This delay is to simulate the air propagation delay. Once the delay is completed, the corresponding timer action, *waveformTimer()*, is triggered in which the waveform entity is processed by customized actions, such as simulating channel collision and passing through multipath channel model. When the timer action has been completed, the waveform entity is forwarded to the output port of the PHY DES module. After the waveform entity has left the DES module, the *waveformExit()* action is called to reset the channel status.

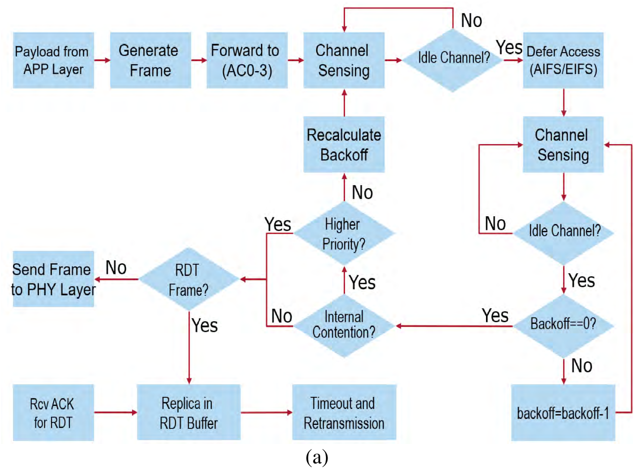
B. MAC LAYER IMPLEMENTATION

Vehicular networking architecture supports vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications. The implementation of a MAC layer should be able to cope with all communication modes. For example, after receiving a frame from the other nodes, the MAC layer should find out whether it comes from another peer vehicle or from infrastructure. Furthermore, the MAC layer should also check the type of the received frame, *i.e.,* broadcast, multicast, or unicast, and prepare for an ACK response to unicast type frames.

1) BRIEF INTRODUCTION TO V2x AND EDCA

One fundamental difference of IEEE 802.11p when compared to other types of IEEE 802.11 networks is the usage of EDCA for the purpose of Quality-of-Service (QoS). Different frames are granted with different priorities. Eight priorities are defined and can be placed in four possible Access Categories (ACs): AC0, AC1, AC2 AC3. Each frame is assigned one of the AC descriptions by the application that created the message depending on the importance and urgency of the content. Specifically, AC0 denotes regular access, AC1 is for non-prior background traffic, while AC2 and AC3 are for prioritized messages, *e.g.,* critical safety messages.

IEEE 802.11 channels are all contention-based, where all nodes need to compete with each other for channel access. During the contention process, the data is required to wait for a random period of time prior to transmitting, which is referred to as *defer access*. The defer access process includes an Arbitration InterFrame Spacing (AIFS), which is a replacement for Distributed coordination function(DCF) InterFrame Space (DIFS), and a backoff period, which is calculated based on a contention window (CW) value. After sensing a busy medium, a node will wait for an AIFS period before sensing the channel again. If the channel is idle, the node will start to backoff, otherwise the node has to wait for another AIFS period. During the backoff period, the node keeps monitoring the channel status. In the event that a busy channel is detected, the node will immediately pause the backoff and restart the AIFS channel sensing step. In short, both AIFS and backoff define the waiting period for a node before accessing the channel.



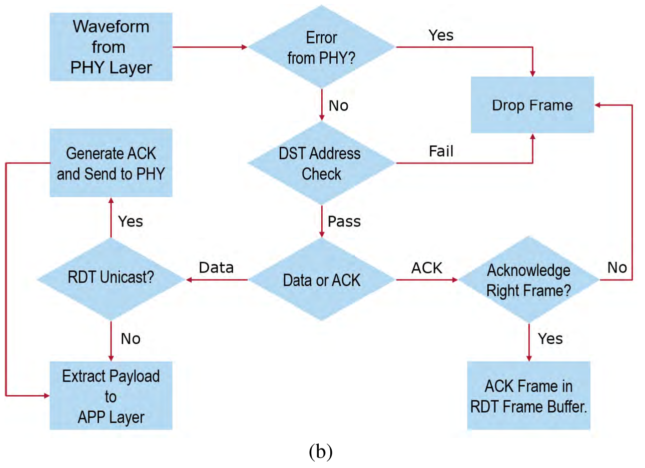


FIGURE 8. Flow chart of EDCA based MAC layer design. MAC outbound indicates the process of receiving a payload from the APP layer and sending a waveform to the PHY layer. MAC inbound is vice versa. (a) MAC layer outbound: Data flow from the APP layer to the PHY layer. A payload is received from the APP layer, converted into a frame, experience channel access backoff and finally converted into a waveform. (b) MAC layer inbound: Data flow from the PHY layer to the APP layer. Drop the corrupt frame entity, or extract the payload from intact frame and send to the APP layer. Reply an ACK frame if necessary.

In EDCA, the ACs decide different (AIFS, backoff) pairs. Therefore, the frames with different priorities own different defer access periods. In general, the higher priority the shorter the defer period and vice versa. The design purpose of EDCA is to enable the frames the with higher priority to gain channel access more frequently.

2) FLOW CHART OF MAC LAYER DESIGN IN DES

In order to depict more clearly the implementation of the MAC layer, we define the data flow from the PHY layer to the APP layer as the inbound flow, as shown in Figure 8(b), and the flow from the APP to the PHY layer as the outbound flow, as shown in Figure 8(a).

For the outbound flow, a payload from the APP layer is converted into a frame by adding the necessary MAC layer headings, then forwarded to the AC0-3 queues. Frames from the four AC queues will perform deferred access simultaneously. It might be possible that more than one of the

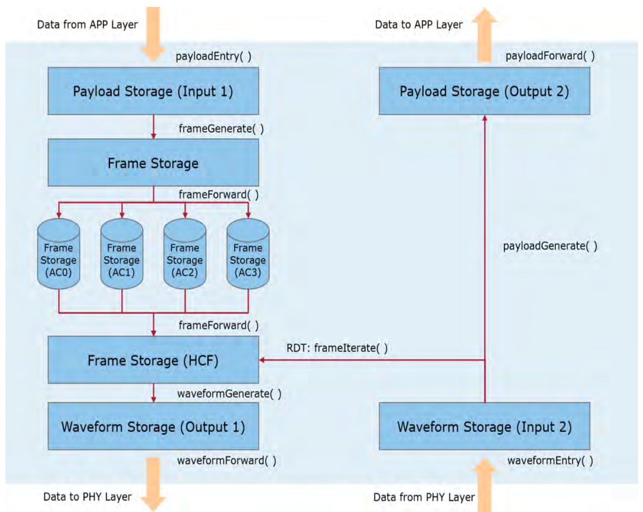


FIGURE 9. Modeling the MAC layer using MATLAB DES. The MAC DES module involves threes types of entities: Payload entity, frame entity and waveform entity. It is responsible for the data streams sending to the PHY layer and receiving from the PHY layer.

AC queues has frames the are ready to send after the deference period, thus a contention is created. Since this situation happens inside the same node, this type of contention is called an *internal contention*, which is unique for nodes using EDCA. Whenever an internal contention occurs, the frame with the highest priority will be the first one to be sent out, while the other frames have to redo the defer access.

If a frame is of the unicast type and requires an ACK from the receiver, *i.e.,* Reliable Data Transmission (RDT), a replica is created inside the buffer and it waits for the ACK. If the ACK is not received within a predefined time period, the replica will be sent again until an ACK is received or the maximum retransmission limit is reached. If the ACK is still not received by then, this frame will be dropped.

For the inbound flow, a waveform is received from the PHY layer. The MAC layer will first check if the frame is intact. The CRC is performed by the PHY Rx, but the action of discarding a corrupted waveform is performed by the MAC layer since the MAC layer is implemented by the MATLAB DES, which is the only option to *destroy* a data entity. If the waveform is intact, the MAC layer needs to check if it is being sent to the correct node by checking the *fromDS/toDS* and *srcAddress/dstAddress* fields.

The type of waveform that could be sent is either a data or an ACK. If it is an ACK, the MAC layer needs to make sure if it is a valid ACK since replicated ACKs in response to the same data may be received due to the congestion of the channel. If it is a data type waveform, the MAC layer extracts the payload and sends it to the APP layer. If it is a RDT waveform, *i.e.,* an ACK is required, the MAC layer will generate the corresponding ACK and send it to the PHY layer.

3) MODELING THE MAC LAYER USING MATLAB DES

Figure 9 illustrates the design of a MAC DES module, in which one payload type storage, six frame type storages, and one waveform type storage are defined to contain payload, frame, and waveform entities, respectively. The *payload type* entity enters into the MAC layer from the APP layer and stays in the payload storage. In the corresponding *payloadEntry() action*, a new event *frameGenerate()* is called, which converts the payload entity to a frame entity by adding the necessary header and trailer.

Depending on its priority, the frame entity is forwarded to different AC queues. Here, the frame entity experiences the channel access deference, backoff, internal contention, and finally is forwarded to a Hybrid Coordination Function (HCF) storage. The *waveformGenerate()* action is triggered in the HCF storage, the frame entity is converted into a waveform entity using the bit level processing technique we introduced in the above section, *i.e.,* PHY (Tx) activities, and finally sent to the PHY layer DES module.

In a reliable data transmission (RDT) environment, a unicast message is required to have an ACK returned. Unlike the broadcast scenario, the frame entity is directly converted to a waveform entity, the unicast frame creates a replica of itself, which is converted into a waveform entity and sent to the wireless channel. The **original** unicast frame entity stays in the HCF storage with a timer attached to it. If the ACK is not received within the timer period, the unicast frame entity creates another replica, converts it to a waveform entity, and retransmits. If the ACK is still not received after the maximum retransmission limit is reached, the unicast frame entity is destroyed in order to prevent further retransmission. If the needed ACK arrives in time, an iteration event is called and its corresponding action *frameIterate()* destroys the original unicast frame in the HCF storage.

The purpose of above design is because in a DES, an event is associated with an entity. If this entity is no longer existing in the DES object, all the associated events become invalid and will never be triggered. Two causes will result in this ‘not existing’ situation. First, an entity is destroyed by event *eventDestroy()*. Second, the entity has left the DES object, *i.e.,* it is forwarded to the output port via event *eventForward(’output’)*. The replica case mentioned above is the second situation. A timer is required for retransmissions, and this timer is a event associated to the unicast frame entity. If this frame entity is converted to a wavefrom entity and forwarded to the PHY Link DES module, the timer will become invalid along with the retransmission activity. Thereby, a replica is necessary to be sent meanwhile the original frame entity stays in the HCF storage with its associated timer activated.

When receiving a waveform type entity from the PHY layer DES module, MAC DES module stores it to the waveform type storage, where the intact waveform is converted to a rxPSDU, the payload is extracted, and the frame sent to the APP layer module. The corrupted waveform entity will be destroyed. If the waveform is an ACK, an iteration event is triggered inside the HCF storage. The original frame entities inside the HCF storage will be iteratively check the sequence number (SN) field until the target frame is found. Then, this frame entity is destroyed along with all its associated events such as retransmission timer events. This indicates the whole reliable data transmission (RDT) process has been completed.

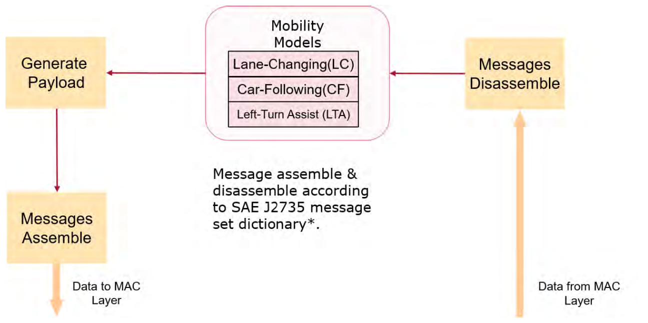


FIGURE 10. **APP Layer Design using MATLAB DES. The messages from mobility model applications are converted into payloads and sent to the MAC layer. When receiving payloads from the MAC layer, the messages are extracted and dispatched to different mobility models.**

C. APP LAYER IMPLEMENTATION

One significant challenge of implementing this proposed simulation model is that the behavior of the APP layer depends on the application itself and there is no comprehensive standard defining all the application requirements since it is almost impossible to anticipate all possible future needs. Therefore, within the scope of this paper, we only focus on the critical safety applications, which will be discussed in Section IV with respect to mobility models. In this section, we only introduce the basic message dissemination functions.

1) MESSAGE DISSEMINATION

A Dedicated Short Range Communication (DSRC) device is required to transmit at least 300 meters [2], [34], and it is assumed that the surrounding vehicle positions are changing frequently in the highly dynamic environment. Consequently, we can assume the safety messages are physically broadcast using a single hop. Therefore, packet collisions and packet loss are major challenges for communication system performance. One solution is to decrease the channel load by grouping similar messages together, as shown in Figure 10.

A mobility model may involve several safety applications, such as lane changing, braking, and collision warning. These applications share different types of messages with other peers differentiated by application IDs (AppIDs). For example, a lane changing application creates messages that include driving direction information, while braking applications generate messages containing brake status. While both types of messages may contain the same information, such as currently location and speed, if these two messages are sent separately the overlapping information will cause a waste of transmission resources. The APP layer DES maintains a message list created by map containers. Whenever an application is activated, it has to register its AppID as well as its message requirements to the message list. The AppID serves as keys while the requirements are values.

Once an empty payload is generated, the APP layer assimilates the data requirements from these applications and

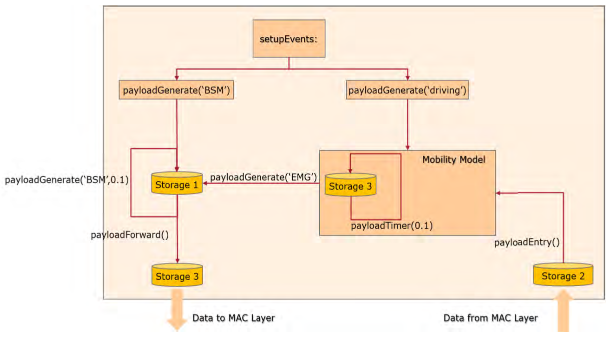


FIGURE 11. Modeling the APP layer using MATLAB DES. The APP DES module generates payload entities containing messages from the mobility models and sends to the MAC DES module. The received messages may update the traffic information in mobility models. The mobility models are integrated and make the vehicles move by the rules.

compiles them into one single message using a dictionary of standardized message construction guidelines. In the last example, an assembled message consisting of position, speed, driving direction, and brake status is created instead of two separate messages. Society of Automotive Engineering (SAE) standards J2735 [35] and SAE J2945 [36] define a dictionary with over 150 data elements. Each data element can be indexed using the AppIDs.

When a payload is received, the APP layer is responsible for separating the data elements according to its appID and dispatches them to the corresponding applications so as to finally affect the mobility models.

2) MODELING THE APP LAYER USING MATLAB DES

Figure 11 shows the design of APP DES module. Since the APP DES is the top layer, it is responsible for generating the needed entities in order to trigger the whole simulation and no external input ports are needed. All entities are generated inside the APP DES module by *setupEvents()* events.

The vehicular mobility models are integrated with the APP layer. This means that the APP DES module has two responsibilities. First, the APP DES module should generate payload entities containing traffic information and sent to the MAC layer. As shown in the figure, two payload storages resource (storage 1 and storage 3) are involved, while the *setupEvents()* generates Basic Safety Messages (BSMs). The 0.1 in *payloadGenerate('BSM',0.1)* event indicates the payload generation intervals since BSMs are generated at a rate of 10 Hz. Once generated, the payload entities are forwarded to Storage 3 and finally sent to the MAC DES Module.

When receiving payloads from the MAC DES module, the APP DES module extracts the message from the payload entities and forwards to the mobility models. The traffic information will be updated according to the received message. A Emergency (EMG) type message can be generated upon request by the vehicles.

The APP DES module is also responsible for making the vehicles to move on the road according to mobility models, thus a new type of entity with a *driving* tag is created and stored in Storage 3. This *driving* entity stays inside Storage 3 forever and will never be sent out. A timer event is recursively triggered on the *driving* entity at an interval of 0.1 seconds. This is the refresh rate of the vehicles moving across the map. The corresponding timer action *payloadTimer(driving)* is called every 0.1 seconds to update the driving information including vehicle position, speed, and direction. This refresh rate can be increased at the cost of execution speed.

IV. MOBILITY MODELS AND SCENARIOS

An integrated vehicular network simulator usually consists of two sections: the vehicular mobility model and the vehicular network model. An accurate vehicular mobility model is necessary for representing the real vehicular traffic behaviors since vehicular mobility significantly impacts the vehicular network performance. A vehicular network application is designed to make use of shared traffic information across the vehicles in order to change traffic patterns, either for the purpose of road safety or for road efficiency improvements.

Therefore, a vehicular network simulator should describe the interactions between the network protocols and vehicular mobility. In the proposed simulation environment, the vehicular mobility models are integrated with the APP layer, where a variety of vehicular mobility models have been proposed for different purposes including random models, flow models, traffic models, behavioral models, and trace-based models. Traffic safety applications usually requires traffic flow modeling, in which the detailed interactions between vehicles are modeled as flows, as shown in Figure 12.

Vehicular network applications can be classified into V2V applications and V2I applications depending on which V2x mode is used. According to [10], V2V applications are generally safety applications while V2I are usually dedicated to traffic efficiency improvements. In this paper, we only focus on V2V communications and only two V2V-based mobility models: (*i*) car following models, and (*ii*) lane changing models.

A. CASE STUDY: V2V COMMUNICATION

Vehicle i is the target vehicle that will perform either car following or lane changing operations. At time t, the x position and velocity of vehicle i are represented as $x_i(t)$ and $v_i(t)$, respectfully. Vehicle $i-1$ and $i+1$ are the vehicles immediately behind and in front of vehicle i with x positions $x_{i-1}(t)$ and $x_{i+1}(t)$, and with speed $v_{i-1}(t)$ and $v_{i+1}(t)$. The variable $\Delta d_i(t)$ indicates the distance from vehicle i to vehicle $i+1$ at time t. For the adjacent lane, the vehicles that are immediately in back and in front are denoted as vehicle $j-1$ and $j+1$. Similarly, their positions and speeds at time t are denoted as $x_{j-1}(t)$ and $x_{j+1}(t)$, as well as $v_{j-1}(t)$ and $v_{j+1}(t)$. These notations are summarized in Table 2.

1) CAR FOLLOWING MODELS (CFMs)

The CFMs control the individual vehicle's driving dynamics in order to maintain a safe distance to the vehicle that is immediately ahead. The objective of CFMs is to model vehicular

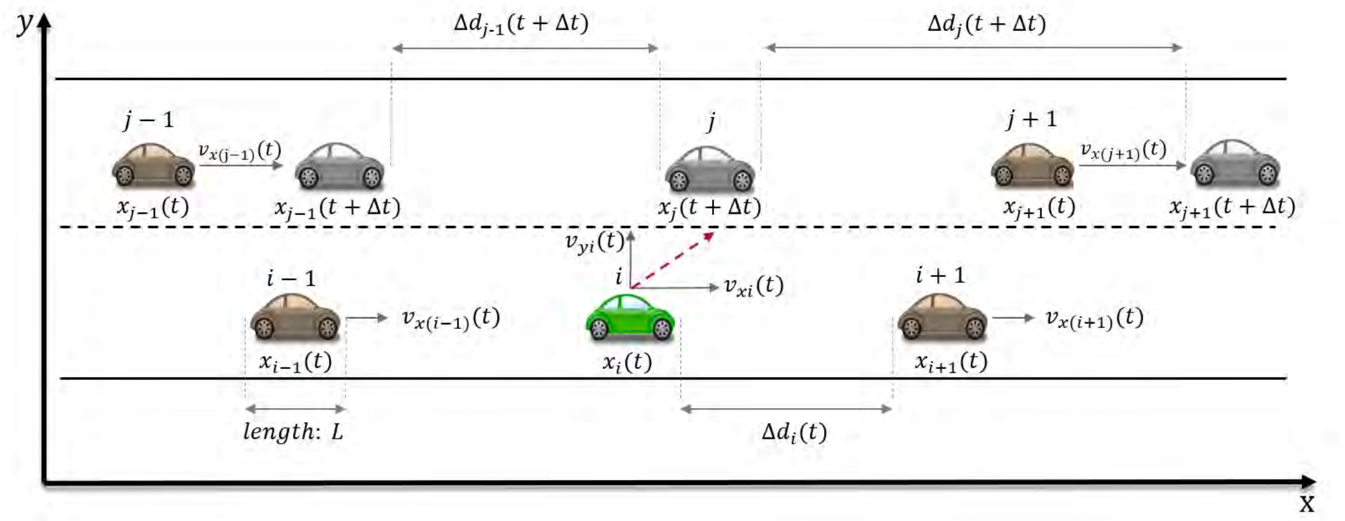


FIGURE 12. Notations for Highway Mobility Model based on V2V Communications. The green car is the target. The green car is surrounded by the yellow cars, who may involve in the coordinated car-following and lane-changing operations. The grey cars are the predicted positions according to the prediction algorithm.

TABLE 2. Notations used in the Highway mobility model.

Symbol	Description
Δt	time step in [seconds]
Δd	distance to the immediately front vehicle in [meters]
acc	acceleration in [m/s^2]
dec	deceleration in [m/s^2]
L	length of vehicle in [meters]
v_x	x velocity in [m/s]
v_y	y velocity in [m/s]
t_r	vehicle reaction period in [seconds]

traffic flows without car collisions using the help of vehicular communications.

$$d_{i,safe} = L_i + t_{react} * v_{xi}(t) + d_{brake}, \tag{8}$$

$$d_{brake} = \frac{v_{xi}(t+\Delta t)^2 - v_{xi}(t)^2}{2 \cdot \mu_s \cdot dec}, \tag{9}$$

$$d_{brake} = \left.\frac{-v_{xi}(t)^2}{2 \cdot dec}\right|_{\mu_s=1, v_{xi}(t+\Delta t)=0}, \tag{10}$$

$$dec_k = -\mu_k * g = -0.8 * 9.8 = -7.84 m/s^2, \tag{11}$$

The safe distance of vehicle i is generally calculated by Eq. (8), where L_i is the length of vehicle i, t_{react} is the reaction time either of the driver or from the autonomous vehicular dynamics, d_{brake} is the braking distance and is calculated in Eq. (9), $v_{xi}(t)$ is the instantaneous speed when brake is performed, and $v_{xi}(t+\Delta t)$ is the velocity when braking action is finished (for a complete stop, $v_{xi}(t+\Delta t) = 0$). Therefore, we have the full stop brake distance defined in Eq. (10).

The deceleration, dec, is determined by the current vehicle speed, road surface friction coefficient μ, as well as the friction type, *i.e.,* static friction and kinetic friction. If a vehicle is driving on a dry concrete road surface, then according to [37] the static friction coefficient is $\mu_s = 1$ and the kinetic friction coefficient is $\mu_k = 0.8$. When the vehicle brakes free and slides, the kinetic friction deceleration dec_k only depends on μ_k and the acceleration due to gravity $g = 9.80\ m/s^2$, as indicated in Eq. (11).

For static friction, NHTSA was able to show a mapping between speed and braking distance, based on which we set the maximum static friction deceleration to $dec_s = -6.50\ m/s^2$ [38]. For $dec \in [dec_s, 0]$, we define this type of braking as *regular brake*. For $dec < dec_s$, the brake action is called as *emergency (EMG) brake*. During EMG brake process, the wheels are drifting on road surface, the friction between them is kinetic friction, thus we set $dec = dec_k$.

In CFMs, the vehicle keeps monitoring the distance to the vehicle immediately ahead of it based on the received BSMs and adjusts its speed adaptively in order to maintain a safe distance. When the front vehicle is braking, the vehicle behind it will be aware of it using the BSM information and it will start to brake.

2) LANE CHANGING MODELS (LCMs)

LCMs are based on multi-lane traffic scenarios. A general LCM should include three parts: the *trigger* for a lane changing event, the *feasibility* of a lane changing event, and the *scheme* used during the lane changing process.

As an example, suppose we consider the situations and notations shown in Figure 12, which describes a basic lane changing scenario involving four vehicles. The *trigger* for vehicle i to change its lane is the distance to vehicle $i+1$ being shorter than the safe distance, *i.e.,* when $\Delta d_i(t) < d_{i,safe}$.

The lane changing *feasibility* of vehicle i is determined by the distance to vehicle $j-1$, *i.e.,* $\Delta d_{j-1}(t+\Delta t)$, and the distance to vehicle $j+1$, *i.e.,* $\Delta d_{j+1}(t+\Delta t)$ during the lane changing process. Both distances should not violate the safe distance rule in order to avoid potential car collisions in the adjacent lane when changing lanes. The speed boundary for vehicle i is calculated as follows:

$$|\frac{0^2 - v_{x,i}(t)^2}{2 \cdot dec_s}| < |\frac{0^2 - v_{x,j+1}(t)^2}{2 \cdot dec_s}| + d_{i,j+1}(t), \tag{12}$$

$$|\frac{0^2 - v_{x,i}(t)^2}{2 \cdot dec_s}| + d_{i,j-1}(t) > |\frac{0^2 - v_{x,j-1}(t)^2}{2 \cdot dec_s}|, \tag{13}$$

$$max(v_{x,i}(t)) = \sqrt{v_{x,j+1}(t)^2 + 2 \cdot |dec_s| \cdot d_{i,j+1}(t)}, \tag{14}$$

$$min(v_{x,i}(t)) = \sqrt{v_{x,j-1}(t)^2 - 2 \cdot |dec_s| \cdot d_{i,j-1}(t)}, \tag{15}$$

The lane changing prediction algorithm assumes the vehicle shifts to the adjacent lane at a y-speed of $v_{y,i}(t)$ while adjusting the x-speed $x_i(t)$ during the process. Eq. (12) calculates the upper speed boundary of vehicle i. During the lane changing process, vehicle $j+1$ can perform a braking operation and the shortest braking distance is determined by dec_s, with $d_{i,j+1}$ being the distance on the x axis before lane changing. The lane changing algorithm should predict whether $d_{i,j+1}$ is a safe distance for a lane change, *i.e.,* $d_{i,j+1}(t)$ should be greater than the difference of the brake trails from both vehicles. Similarly, the lower speed boundary is calculated by Eq. (13). The maximum and minimum speeds for vehicle i during a lane changing operation are defined by Eq. (14) and (15), respectively.

If the lane changing feasibility is not fulfilled, then vehicle i will need to reduce its speed $v_{x,i}(t)$, *i.e.,* brake, in order to meet the safety constraint. If the feasibility check does allow for a lane change, vehicle i will start to change lanes. The trajectory model includes a lane changing period, target lane chosen, *etc*. The validation and performance evaluation of the above mobility models have been presented in [39].

More sophisticated CFMs and LCMs have been proposed, including the Intelligent Driver Model (IDM) [40], Wiedemann Model [41], the Nagel-Schreckenberg model [42], and the Krauss (1998) Model [43]. These models can be implemented and used in *VANET Toolbox* if necessary.

V. PERFORMANCE ANALYSIS

In this section, the performance of the proposed simulation environment is evaluated. We first discuss the computational costs of a full-stack vehicular network simulator in terms of the number of events scheduled during the simulation. Then, we compare the packet success rate (PSR) of BPSK in a AWGN channel between MATLAB PHY layer implementation and NS-3 error rate model. Due to the bit level processing of the PHY layer, the channel tracking (CT) techniques can be enabled on the L-LTF field in order to cope with the high Doppler spread. The performance of CT is evaluated for BPSK signal with different channel environment. Furthermore, based on the case study of V2V communication in the above section, we perform two sets of simulations focusing on the MAC layer behaviors and compare the performance of EDCA with distributed coordination function (DCF).

A. PERFORMANCE OF VANET TOOLBOX

In this section, the performance of *VANET Toolbox* is analyzed and evaluated in terms of events numbers and the execution time with different number of vehicles.

1) COMPUTATIONAL COSTS IN TERMS OF EVENTS

A detailed simulation of the entire vehicular network stack is time-consuming, especially with a large number of vehicles to simulate and taking into consideration large-scale vehicular communication effects. In this section, we show the computational costs in term of the number of events E in a discrete event-based simulation model.

Suppose we have n vehicles in a vehicular communication scenario, with each vehicle transmitting data at rate r in Hertz, and the simulation time is t in seconds. When a vehicle is willing to send a message, events are scheduled in order to generate the message in the APP layer and forwarded to the MAC layer, where the message is converted to a frame, experiences channel sensing, backoff, and finally sent to the PHY layer, all of which are conducted by different events. In the PHY layer, the frame is transformed into a waveform and sent into the wireless channel by events. After receiving a waveform, events are called in order to extract the information from the waveforms and send it way up to the APP layer and process it in the mobility models. We assume the number of events E per transmission is e. The number of events E per simulation can be calculated by :

$$E(n, r, t) = (nr) \cdot e \cdot (n-1) \cdot t, \tag{16}$$

It is observed that the number of events E is linearly proportioned with the simulation time t and the data rate r, but nonlinear with respect to the number of vehicles n. Figure 13 shows the number of simulated events for each layer of the vehicular network and for the overall communication set. We choose the car following model (CFM) as the scenario since for this CFM each vehicle broadcasts only BSMs at a rate of 10 Hz and no other transmissions are involved. For a 600-second simulation when 4 vehicles are involved, the total number of events is around 2×10^6. When the number of vehicles increases to 36, the total number of events is around 1.5×10^8, which represents an increase by a factor of 75.

In addition to the computational cost in terms of number of events, we also profile the execution time for a 30-second simulation with 30 vehicles. The overall execution time costs 8535 seconds, among which the PHY Tx function *phy_psdu2waveform* consumes 511 seconds (5.98%) and the PHY Rx function *phy_waveform2psdu* consumes 4337 seconds (50.7%). These two functions include all bit-level processing operations of the PHY layer. As we have

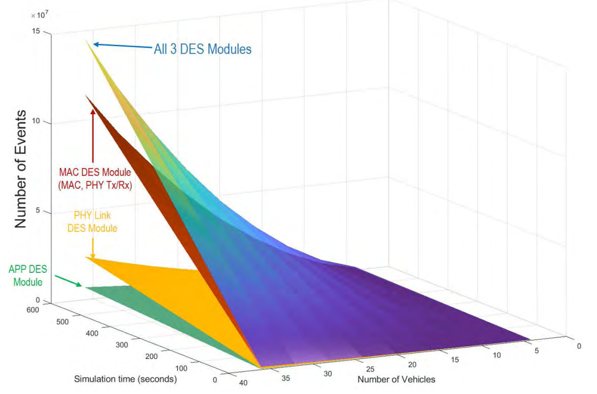


FIGURE 13. Computational cost in terms of events for each layers. The mono-color surfs indicate the number of events happened for the PHY layer, the APP layer and the MAC layer (from bottom to top). The colorful surf is the combination of events happened in all three layers.

introduced in Section III-A, the PHY Tx function is straightforward, *i.e.,* serializing the PSDU to binary bits. While most of the bit-level processing operations such as packet detection, channel estimation and CRC verification are in the PHY Rx function. Thus, the PHY Rx function costs more execution time than the PHY Tx function. Additionally, the result also shows that even though the MAC layer creates the largest portion of events, the bit-level PHY layer costs more computational cost in terms of execution time.

Among all the layers, the APP layer costs the fewest number of events since it is the top layer of the network stack and it mainly deals with message generation and reception. In our case, the BSM is the only application data generated in the APP DES. If additionally applications are involved or the data generation rate is increased, the number of events will be increased accordingly. The PHY link DES involves slightly larger number of events relative to the APP layer because whenever the APP DES generates one message and when this message enters the PHY link DES module in the format of a waveform entity, it triggers a series of events to deal with activities such as delay, buffer, and waveform check. Therefore, the number of events in the PHY link DES correlates to the number of messages generated in the APP DES module. The number of events increases dramatically in the MAC DES module. This is because the number of events in the MAC DES module is not only correlated to the number of APP layer messages but also affected by the channel status. Suppose if the wireless channel is congested, the channel sensing operation would be performed more frequently in order to monitor the channel status and seek a transmission opportunity. Consequently, the timer event related to the channel sensing operation is called more frequently, which might cause a burst amount of events in the MAC DES module. Additionally, since the PHY Tx and Rx functions are integrated with the MAC/PHY DES Module, the events caused by PHY Tx/Rx are shown in the MAC/PHY DES module surf instead of PHY Link DES module surf.

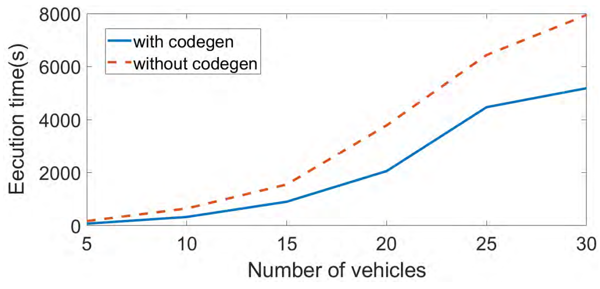


FIGURE 14. Execution speed improvement due to MATLAB code generation. The simulation time is 30 seconds, the number of vehicles increases from 5 to 30.

2) SimEvents CODE GENERATION

MATLAB is a high-level interpreted type language and provides an interactive programming environment. Compare with lower-level languages such as C/C++, the execution time is longer. Furthermore, the PHY layer includes bit-level processing, which uses more numerical computations so that the execution speed is further slowed down.

In order to enhance the execution speed, MATLAB/Simulink supports converting MATLAB code into C/C++ code using code generation techniques. Figure 14 presents a comparison of the execution time with the code generation (codegen) feature enabled and disabled. The computational testbed possesses the following characteristics: i7-6700k at 4.0GHz (CPU), 32G DDR4 2133MHz (memory),Microsoft Windows 10 (OS).

In this figure, the execution time increases as the number of vehicles increases. The red line indicates the simulations running in *interpreted execution* mode, *i.e.,* without code generation. The blue line is the execution time with code generation. Based on this figure, the code generation shows its advantage with respect to accelerating the execution speed. For the simulation with 30 vehicles, *interpreted execution* costs 7945 seconds, while *code generation execution* costs 5185 seconds. The execution speed of 30 vehicles simulation increases 53.23% because of code generation.

B. PERFORMANCE OF THE PHY LAYER

In this section, the performance of the MATLAB PHY layer is evaluated and compared with the NIST error rate model of NS-3, which is broadly adopted by iTETRIS and VSimRTI projects.

1) PRECISE PHY LAYER MODELING

NS-3 is a packet-based, discrete-event network simulator equipped with several wireless models. When a waveform is received, NS-3 calculates the signal to noise ratio (SNR) and invokes its error rate model to decide the packet successful reception rate. Two error rate models are integrated with NS-3: the YANS [44] model and the NIST [45] model. The YANS model, which is based on an analytical bound was replaced with NIST error model in 2010. In this paper, we only focus on the currently used NIST error rate model.

In order to estimate the Packet Success Rate (PSR) for orthogonal frequency division multiplexing (OFDM) symbols, NIST calculates $\frac{E_b}{N_o}$ (SNR per bit E_b to the one-side noise spectral density N_o) based on SNR in dB using :

$$\frac{E_b}{N_o} = SNR - 10log10(k), \tag{17}$$

where $k = log_2(M)$, M is the modulation level, and k is the number of bits per symbol. However, the NIST error model has two limitations.

First, the NIST error model does not consider the oversampling situation and equates SNR to $\frac{E_s}{N_o}$, *i.e.,* ratio of symbol energy to noise power spectral density. The relationship between $\frac{E_s}{N_o}$ and SNR in dB for complex signals is defined by [46]:

$$\frac{E_s}{N_o} = SNR + 10log_{10}\frac{T_{sym}}{T_{samp}}, \tag{18}$$

where T_{sym} is the symbol period of the signal and T_{samp} is the sampling period of the signal. For a complex baseband signal, if it is oversampled by a factor of n, then $\frac{E_s}{N_0}$ does not equal to SNR but exceed by $10log_{10}(n)$.

Second, the NIST error model does not account for the energy in nulls. Take IEEE 802.11p for instance, an OFDM signal consists of 64 subcarriers, among which 48 subcarriers are for data, 4 subcarriers for pilot information and 12 subcarriers are NULL. Thus, the SNR for occupied subcarriers in dB, SNR_o is calculated using :

$$SNR_o = SNR - 10log_{10}\frac{N_{FFT}}{N_{data} + N_{Pilot}}, \tag{19}$$

where N_{FFT} is the number of FFT sampling points, *i.e.,* the total number of subcarriers for a OFDM signal. N_{data} is the quantity of subcarriers used for data and N_{pilot} is for pilot.

Figure 15 shows the comparison of PSR on AWGN channels between NIST error model and proposed MATLAB error model. The PSR of the NIST error model is over optimistic while the proposed MATLAB error model is more realistic. As a packet-based network simulator, the NIST model is the most comprehensive NS-3 model can implement. The oversampling situation, as well as the energy in the null subcarriers, requires processing of the bit level, thus it is potentially challenging for NS-3 to implement these features. The AWGN channel shown in Figure 15 is a simple scenario that can be compensated by incorporating an offset to NS-3 simulator. However, in a more complicated environment, such as Non-line-of-sight (NLOS) conditions in an Urban scenario, a constant offset could potentially be insufficient to cope with the different channel models.

2) PHY LAYER ON BIT-LEVEL PROCESSING

Another limitation of NS-3 is the oversimplified PHY layer. The packets are forwarded among objects of *Packet class* via methods. Based on the packet-error rate (PER) obtained from *NIST_Error_Model*, NS-3 randomly corrupts received packets in order to emulate the packet corruption process. However a realistic PHY layer of a communication system consists of more functions including frequency offset correction, channel estimation, modulation, and demodulation. The proposed simulator implements all the PHY layer features at the bit level.

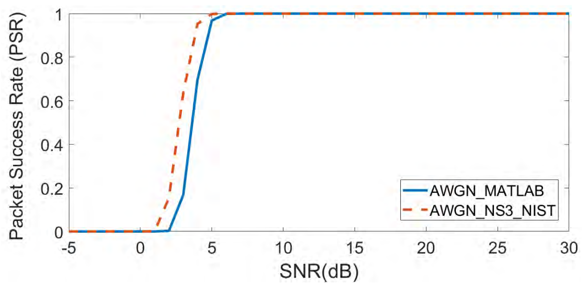


FIGURE 15. The packet success rate (PSR) comparison between MATLAB and NS-3 (NIST model). The simulation is conducted with BPSK modulation in AWGN channel.

In a vehicular network environment, the V2x channels have different characteristics compared with other stationary indoor channels [47]. First, V2x channels are affected by longer multipath fading, which increases the possibility of intersymbol interference (ISI). Second, the transmission environment is highly dynamic, which causes significant Doppler effects resulting in more channel fading. When passing through the V2x channels, the waveforms are impacted more than just passing through an AWGN channel. The performance in terms of PSR will be degraded, thus channel tracking techniques are needed in order to enhance the performance.

In the proposed simulator, we integrate a time and frequency selective multipath Rayleigh fading channel as specified by [48] with an AWGN channel. The conventional WLAN channel estimation from L-LTF is used for the entire packet duration. In order to compensate the high Doppler spread of the V2x channel, channel tracking is enabled. With channel tracking, the channel estimation obtained from L-LTF is updated per symbol using decision directed channel tracking as presented in [47]. We compare the performance in terms of PSR on BPSK across different scenarios including Highway LOS and Urban NLOS with channel tracking (CT) on and off. The results are presented in Figure 16. In this figure, the receiver with channel tracking (CT) enabled possesses a better PSR in V2x channels. Due to the restrictions of NS-3 on packet-level processing, implementing channel tracking for OFDM symbols is relatively difficult to perform. It is worth mentioning the focus of this paper is the presentation of the ability and accuracy of bit-level processing, and using simple channel models can help in providing a clearer evaluation for straightforward comparison. Nevertheless, the authors will explore more complicated channel models in future research activities.

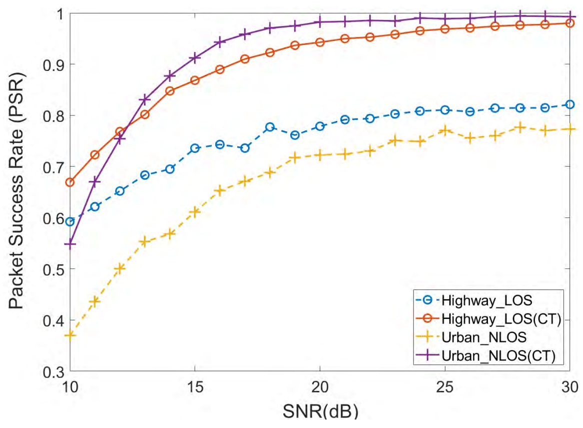


FIGURE 16. The performance of channel tracking (CT) for BPSK modulation in multi-path fading channel. The channel models involves highway line-of-sight (LOS) and urban non-line-of-sight (NLOS).

TABLE 3. Parameters of CSMA and EDCA in the simulations.

Parameters	CSMA (802.11a)	EDCA (802.11p)
IFS	2	3 (AC2) & 2(AC3)
SIFS (μs)	16	32
slot time (μs)	9	13
[CWmin,CWmax]	[15,1023]	[7,15] (AC2), [3,7] (AC3)

C. PERFORMANCE COMPARISON BETWEEN EDCA AND DCF

One significant feature of a vehicular network is adopting EDCA in the MAC layer. In this section, we will compare the performance between EDCA and distributed coordination function (DCF), *i.e.,* carrier sensing media access (CSMA), where the latter is generally used in other IEEE 802.11 products. By analyzing the simulation results, we can evaluate the effectiveness of the proposed simulator.

The simulation scenario is a unidirectional highway consisting of two lanes. The car following model (CFM) is chosen to be the mobility model. The vehicles are broadcasting BSMs (AC2) at 10 Hz and AC3 messages using a Poisson distribution modeled by $\lambda = 2$. We performed two sets of simulations for EDCA and CSMA with the key MAC parameters listed in Table 3. As the PHY layer of IEEE 802.11p is derived from IEEE 802.11a, we choose the IEEE 802.11a version of CSMA to minimize the difference with other layers.

Figure 17 shows the simulation results of the Packet Deliver Latency (PDL) and Packet Deliver Rate (PDR). As all the messages in the simulation are broadcast, there are no retransmissions involved, with the latency mainly coming from the channel access deference process, *i.e.,* IFS+backoff. In Table 3, the Inter-Frame-Space (IFS) of CSMA, *i.e.,* DIFS, equals to the highest priority IFS (AC3), *i.e.,* AIFSN(AC3), and slightly smaller than AIFSN (AC2). The Shortest-IFS (SIFS) and slot time of IEEE 802.11p is greater than IEEE 802.11a in order to cope with the mobility characteristics of

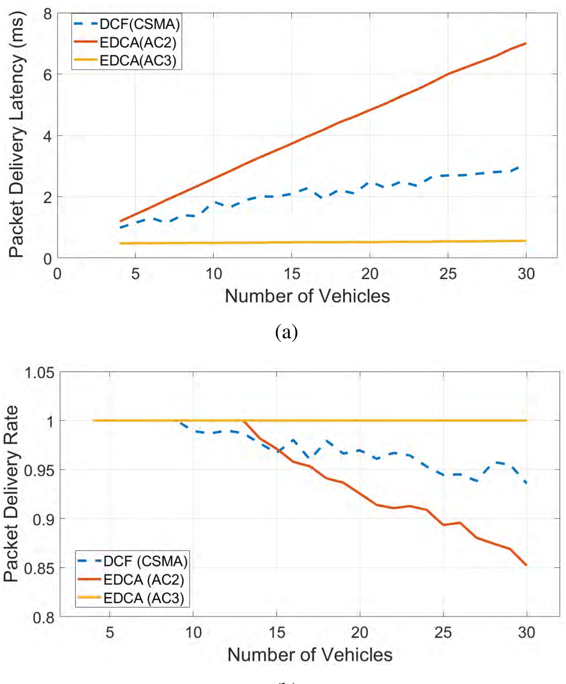


FIGURE 17. Performance Evaluation of DCF (CSMA) and EDCA (AC2-3) on packet delivery latency (PDL) and packet delivery rate (PDR) as the density of vehicle increases. (a) Packet Delay of EDCA (AC2), EDCA (AC3) and DCF (CSMA). Delay mainly comes from the backoff process during the channel access deference stage since no retransmissions are required for broadcast packets. (b) Packet Delivery Rate (PDR) of EDCA (AC2), EDCA (AC3) and DCF (CSMA). Packet corruption comes from either packet collision in the channel or severe noise due to the low SNR.

the vehicular network. Only a minimum Contention Window (CW) is used for broadcasting purpose.

In Figure 17(a), the latency of CSMA is smaller than EDCA (AC2) but greater than EDCA (AC3). This is due to the fact that data possessing different priorities are queued into different ACs, while in CSMA all data are buffered in the same queue. Whenever an internal collision happens, AC2 always gives way to AC3. This is why AC3 shows a steady and better performance in the figure. According to SAE J2735 [35], the maximum latency for safety messages is 10 ms. When the latency of AC2 is below the threshold, EDCA is shown to be the better option than CSMA.

Fig. 17(b) compares the Packet Delivery Rate (PDR) of CSMA with AC2 and AC3 transmissions of EDCA. When the number of vehicles increases, the PDR of AC3 maintains at nearly 100 percent. This proves the AC2 and AC3 can coexist in the same channel, and AC2 traffic affects little on AC3 traffic. On the other hand, the PDR of AC2 starts to decrease when number of vehicles approaches to 15 due to the packet collisions. When less than 15 vehicles, EDCA (AC2) still performs better than CSMA. However, when more than 15 vehicles are present, CSMA acts better than EDCA (AC2). This is due to the coexistence of AC2 and AC3, which increases the packet collision rate. However, for 30 vehicles, the PDR of EDCA (AC2) is till above 85 percent, which performs well enough on broadcasting BSMs.

TABLE 4. Comparisons of features and limitations between proposed simulator and NS-3.

	VANET Toolbox	NS-3
Features	1. Accurate bit-level PHY layer simulation. 2. Expandable to SDR hardware and other toolboxes. 3. Support traffic simulation in real time. 4. Support multi-OS: Windows, Linux, OSX.	1. Full stack simulation on varieties of network types. 2. Well developed comprehensive simulation modules. 3. Relative fast simulation speed. 4. Free software, open source
Limitations	1. Parameters are fixed during simulation. 2. Relatively slow simulation speed. 3. MATLAB/Simulink and toolboxes are not free. 4. Still under development, bugs may exist.	1. Packet-based PHY layer, less accurate, 2. Limited hardware radio expandability. 3. Limited support on real-time traffic simulation. 4. Primary running in Linux, less supports in other OS.

D. COMPARISONS BETWEEN PROPOSED SIMULATOR AND NS-3

NS-3 is a well known discrete-event network simulator that supports full stack standards across a variety of networking applications. The NS-3 modules are robust and can operate at a faster speed due to its C++ implementation. However, NS-3 potentially possessed several limitations. First, its PHY layer is packet-based, which means the minimum data element is at the packet level instead of the bit level. Therefore, the bit-related operations such as channel tracking, channel estimation, and frequency offset correction cannot be applied. Additionally, NS-3 lacks supports for real radio hardware as it is unable to convert information into bits or symbols. Second, NS-3 was originally designed to operate in a pure network simulation environment. In order to simulate vehicular traffic, NS-3 either uses predefined route information or interacts with mobility simulators asynchronously with the help of interfaces. The randomness of vehicular traffic scenario might not be able to be simulated in real-time.

Our proposed simulation environment compensates for the limitations of NS-3. First, it provides a more accurate PHY layer representation at the bit level. The simulator is able to model the channel impairments such as noise, path loss, or shadow fading on the bits. Moreover, the bits are converted into symbols, which are exactly the same format in the real wireless communications. Thus, it is reasonable to infer the simulated wireless channel can be replaced by actual software defined radios (SDR) such as the USRP. Real radio transmissions will be evaluated in future research activities of the authors. Furthermore, as the mobility models are integrated with the APP layer, the reciprocal interactions between the traffic application and the network communication are sufficiently supported. This feature makes the proposed simulator to simulate vehicular driving operations and network communications sufficiently synchronized in real time.

On the other hand, the proposed simulator has several limitations. First, MATLAB is an interpreted programming language that aims for precision modeling. Compared with other compiled programming languages such as C++, which is used in NS-3, fast execution speeds or low computational costs are often difficult to achieve in MATLAB, especially with the PHY layer of *VANET Toolbox*, which is designed on bit level processing. It turns out that the proposed implementation is significantly slower than NS-3. Furthermore, *VANET Toolbox* does not support parallel computing, *i.e.,* the model created by *VANET Toolbox* cannot be operated across multiple cores. Even though *SimEvents* supports C/C++ code generation since MATLAB R2017b, it still has several limitations. For instance, C/C++ code generation does not support hash map, persistent variables, or changing the values of properties of an object inside another object. Those functions have to be declared as extrinsic functions and cannot enjoy the benefit of C/C++ code generation. Thus, the relatively slow execution speed is the major limitation of the proposed simulator. Second, the movements of entities among different DES modules require the support of Simulink. Therefore, the *VANET Toolbox* inherits the limitations of Simulink. For example, parameters such as the total number of vehicles cannot be changed during the simulation since Simulink locks all the parameters. Nevertheless, all of these limitations may be solved in future releases of MATLAB/Simulink. Table 4 summarized the major features and limitations for both simulators.

VI. CONCLUSION

In this paper, we presented an integrated vehicular network simulator called the *VANET Toolbox* that is based on MATLAB Discrete Event System (DES). This is the first vehicular network simulator in the MATLAB/Simulink environment that supports a full stack of network protocols. The design structure of the main components namely, the APP layer, MAC layer, PHY layer, and basic mobility models, were proposed and were demonstrated to accurately simulate the inter-communications between vehicles in different scenarios but not without limitations. The *VANET Toolbox* requires only MATLAB/Simulink to use it in addition to *SimEvents*, *Communication and WLAN System Toolbox*. The design purpose of the *VANET Toolbox* is to provide a framework and an opportunity to attract more researchers to improve it and finally benefit the vehicular network development. The vehicular network simulator, *VANET Toolbox*, is open source and can be downloaded from https://github.com/lewangwpi/vanet_toolbox or https://www.mathworks.com/matlabcentral/fileexchange/684 37-vanet-toolbox-a-vehicular-network-simulator-based-on-des.

ACKNOWLEDGMENT

The authors would like to thank the generous support of the MathWorks and the Brazilian Government Science Without Borders program. Furthermore, they would also like to acknowledge the guidance and input for Mike McLernon of the MathWorks throughout these research activities.

REFERENCES

[1] M. Müller, "WLAN 802.11 p measurements for vehicle to vehicle (V2V) DSRC," in *Application Note Rohde Schwarz*, vol. 1, pp. 1–25, 2009.

[2] J. B. Kenney, "Dedicated short-range communications (DSRC) standards in the United States," *Proc. IEEE*, vol. 99, no. 7, pp. 1162–1182, Jul. 2011.

[3] H. Hartenstein and L. P. Laberteaux, "A tutorial survey on vehicular ad hoc networks," *IEEE Commun. Mag.*, vol. 46, no. 6, pp. 164–171, Jun. 2008.

[4] M. Chowdhury, "15th intelligent transportation systems world congress," *J. Intell. Transp. Syst.*, vol. 14, no. 2, pp. 51–53, 2010. doi: 10.1080/15472451003738194.

[5] F. Ahmed-Zaid, H. Krishnan, M. Maile, L. Caminiti, S. Bai, and S. VanSickle, "Vehicle safety communications—Applications: System design & amp, objective testing results," *SAE Int. J. Passenger Cars Mech. Syst.*, vol. 4, pp. 417–434, Jun. 2011.

[6] G. Howe, G. Xu, D. Hoover, D. Elsasser, and F. Barickman, "Commercial connected vehicle test procedure development and test results–emergency electronic brake light," Tech. Rep., 2016.

[7] G. Howe, G. Xu, D. Hoover, D. Elsasser, and F. Barickman, "Commercial connected vehicle test procedure development and test results–blind spot warning/lane change warning," Tech. Rep., 2016.

[8] G. Howe, G. Xu, D. Hoover, D. Elsasser, and F. Barickman, "Commercial connected-vehicle test procedure development and test results–intersection movement assist," Tech. Rep., 2016.

[9] J. Harri, F. Filali, and C. Bonnet, "Mobility models for vehicular ad hoc networks: A survey and taxonomy," *IEEE Commun. Surveys Tuts.*, vol. 11, no. 4, pp. 19–41, Dec. 2009.

[10] H. Hartenstein and K. Laberteaux, *VANET: Vehicular Applications and Inter-Networking Technologies*, vol. 1. Hoboken, NJ, USA: Wiley, 2009.

[11] D. O. Lazaro, E. Robert, L. Lan, J. Gozalvez, S. Turksma, F. Filali, F. Cartolano, M. A. Urrutia and Krajzewicz, "An integrated wireless and traffic platform for real-time road traffic management solutions," in *COMeSafety Newsletter*, 2010.

[12] L. Bononi, M. Di Felice, G. D'Angelo, M. Bracuto, and L. Donatiello, "MoVES: A framework for parallel and distributed simulation of wireless vehicular ad hoc networks," *Comput. Netw.*, vol. 52, no. 1, pp. 155–179, 2008.

[13] M. Killat, F. Schmidt-Eisenlohr, H. Hartenstein, and C. Rössel, P. Vortisch, S. Assenmacher, and F. Busch, "Enabling efficient and accurate large-scale simulations of VANETs for vehicular traffic management," in *Proc. 4th ACM Int. workshop Vehiculr Ad Hoc Netw.*, Sep. 2007, pp. 29–38.

[14] S.-Y. Wang, C. L. Chou, C. H. Huang, C. C. Hwang, Z. M. Yang, C. C. Chiou, and C. C. Lin, "The design and implementation of the NCTUns 1.0 network simulator," *Comput. Netw.*, vol. 42, no. 2, pp. 175–197, 2003.

[15] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO-Simulation of Urban MObility," *Int. J. Adv. Syst. Meas.*, vol. 5, nos. 3–4, pp. 128–138, Dec. 2012.

[16] Nsnam. (Feb. 2017). *NS-3 Project-Introduction to NS-3*. [Online]. Available: https://www.nsnam.org/docs/tutorial/html/introduction.html

[17] G. Pongor, "Omnet: Objective modular network testbed," in *Proc. Int. Workshop Modeling, Anal., Simulation Comput. Telecommun. Syst.*, 1993, pp. 323–326.

[18] A. F. G. Gil, *Traci4matlab: User's Manual*. Universidad Nacional De Colombia, 2014.

[19] C. Gorgorin, V. Gradinescu, R. Diaconescu, V. Cristea, and L. Ifode, "An integrated vehicular and network simulator for vehicular ad-hoc networks," in *Proc. 20th Eur. Simulation Modelling Conf.*, vol. 59, May 2006, pp. 1–8.

[20] M. R. Stiglitz and C. Blanchard, *IEEE Standard Dictionary of Electrical and Electronics Terms*. 1992, Sec. 4.

[21] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Springer, 2009.

[22] T. K. Som and R. G. Sargent, "A probabilistic event scheduling policy for optimistic parallel discrete event simulation," in *Proc. 12th Workshop Parallel Distrib. Simulation*, May 1998, pp. 56–63.

[23] K. Akesson, M. Fabian, H. Flordal, and R. Malik, "Supremica-an integrated environment for verification, synthesis and simulation of discrete event systems," in *Proc. 8th Int. Workshop Discrete Event Syst.*, Jul. 2006, pp. 384–385.

[24] R. Davidrajuh, "Developing a new Petri net tool for simulation of discrete event systems," in *Proc. 2nd Asia Int. Conf. Modelling Simulation (AMS)*, May 2008, pp. 861–866.

[25] C.-H. Chen, "A hybrid approach of the standard clock method and event scheduling approach for general discrete event simulation, " in *Proc. 27th Conf. Winter simulation*, Dec. 1995, pp. 786–790.

[26] S. Zhongyuea and G. Zhongliang, "Vehicle routing problem based on object-oriented discrete event simulation," in *Proc. 2nd Int. Conf. Adv. Comput. Control*, vol. 5, Mar. 2010, pp. 638–643.

[27] *IEEE Standard for Information Technology–Telecommunications and Information Exchange Between Systems Local and Metropolitan Area Networks–Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, Standard, Dec. 2016.

[28] *Vehicles in Network Simulation (VeinS)*. [Online]. Available: http://veins.car2x.org

[29] *i TETRIS: The Open Simulation Paltform for Intelligent Transport System (ITS) Services*. [Online]. Available: http://www.ictitetris.eu/itetris_platform.html

[30] *VSimRTI—Smart Mobility Simulation*. [Online]. Available: https://www.dcaiti.tuberlin.de/research/simulation/

[31] S. Papanastasiou, J. Mittag, E. G. Strom, and H. Hartenstein, "Bridging the gap between physical layer emulation and network simulation," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Apr. 2010, pp. 1–6.

[32] B. Aygun, "Distributed adaptation techniques for connected vehicles," Ph.D. dissertation, WPI, Worcester, MA, USA, 2016.

[33] Mathworks. (May 2017). *802.11n Packet Error Rate Simulation for 2x2 TGn Channel*. [Online]. Available: https://www.mathworks.com/help/wlan/examples/802-11n-packet-error-rate-simulation-for-2x2-tgn-channel.html

[34] C. Michaels, "DSRC implementatio guide–a guide to users of SAE J2735 message sets over DSRC," SAE, Pennsylvania, PA, USA, Tech. Rep., 2010.

[35] *J2735 Dedicated Short Range Communications (DSRC) Message Set Dictionary*, Standard, SAE.

[36] *J2945 On-Board System Requirements for V2V Safety Communications*, Standard, SAE.

[37] *Static and Kineticfriction*. [Online]. Available: http://ffden-2.phys.uaf.edu/211_fall2002.web.dir/ben_townsend/staticandkineticfriction.htm

[38] NHTSA. (Sep. 1999). *Federal Motor Vehicle Safety Standards; Stopping Distance Table*. [Online]. Available: https://www.gpo.gov/fdsys/pkg/FR-1999-09-07/pdf/99-23226.pdf

[39] L. Wang, R. F. Iida, and A. M. Wyglinski, "Coordinated lane changing using v2v communications," in *Proc. IEEE 88th Vehicular Technol. Conf. (VTC-Fall)*, Aug. 2018, pp. 1–5.

[40] A. Kesting, M. Treiber, and D. Helbing, "Enhanced intelligent driver model to access the impact of driving strategies on traffic capacity," *Philos. Trans. Roy. Soc. A, Math., Phys. Eng. Sci.*, vol. 368, no. 1928, pp. 4585–4605, 2010.

[41] B. Higgs, M. M. Abbas, and A. Medina, "Analysis of the wiedemann car following model over different speeds using naturalistic data," in *Proc. Procedia RSS Conf.*, 2011, pp. 1–22.

[42] A. Schadschneider, "The nagel-schreckenberg model revisited," *Eur. Phys. J. B, Condens. Matter Complex Syst.*, vol. 10, no. 3, pp. 573–582, 1999.

[43] S. Krauß, "Towards a unified view of microscopic traffic flow theories," *IFAC Proc. Volumes*, vol. 30, no. 8, pp. 901–905, 1997.

[44] M. Lacage and T. R. Henderson, "Yet another network simulator," in *Proc. workshop ns-2, IP Netw. simulator*, Oct. 2006, p. 12.

[45] G. Pei and T. R. Henderson, "Validation of OFDM error rate model in ns-3," in *Boeing Research Technology*, 2010, pp. 1–15.

[46] MathWorks. *AWGN Channel*. [Online]. Available: https://www.mathworks.com/help/comm/ug/awgnchannel.html

[47] J. A. Fernandez, D. D. Stancil, and F. Bai, "Dynamic channel equalization for IEEE 802.11p waveforms in the vehicle-to-vehicle channel," in *Proc. 48th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep./Oct. 2010, pp. 542–551.

[48] P. Alexander, D. Haley, and A. Grant, "Cooperative intelligent transport systems: 5.9-GHz field trials," *Proc. IEEE*, vol. 99, no. 7, pp. 1213–1235, Jul. 2011.

LE WANG received the M.Sc. and Ph.D. degrees in electrical and computer engineering from Worcester Polytechnic Institute, Worcester, MA, USA, in 2013 and 2019, respectively. For the past four years, he has been collaborating with MathWorks, Inc., on designing vehicular network simulation environment using MATLAB discrete-event system. His current research interests include wireless communication, vehicular network simulation, wireless network security, and applications development based on V2x communication. He has been a member of the Wireless Innovation Laboratory, since 2011.

RENATO IIDA received the B.Eng. degree in telecommunications engineering from the University of Brasilia, Brazil, in 2002, and the M.Sc. degree in electrical engineering from the University of Brasilia, in 2006. He is currently pursuing the Ph.D. degree in electrical engineering with the Worcester Polytechnic Institute, Worcester, MA, USA. From 2008 to 2014, he worked in system level simulations with Nokia, and was involved in the standardization of MUROS, VAMOS feature in GSM network in 3GPP. His current research interests include V2V networks and to improve the radio resource management in the type of networks. He received an award from CAPES with science without borders program to support his Ph.D.

ALEXANDER M. WYGLINSKI received the B.Eng. and Ph.D. degrees from McGill University, Montreal, QC, Canada, in 1999 and 2005, respectively, and the M.Sc. (Eng.) degree from Queen's University, Kingston, ON, Canada, in 2000, all in electrical engineering. He is currently a Professor of electrical and computer engineering and a Professor of robotics engineering with Worcester Polytechnic Institute, Worcester, MA, USA, where he is currently the Director of the Wireless Innovation Laboratory. Throughout his academic career, he has published over 35 journal papers, over 80 conference papers, nine book chapters, and two textbooks. His current research interests include wireless communications, cognitive radio, software-defined radio, dynamic spectrum access, spectrum measurement and characterization, electromagnetic security, wireless system optimization and adaptation, and cyber-physical systems.