Journal of Harbin Engineering University
ISSN: 1006-7043

Vol 45 No. 3
March 2024

Network Simulation Environment for Visible Light Communication in Vehicular Networks

Boris Tomaš[1],Neven Vrček[2]

[1]School of Computer Science and Statistics, Trinity College Dublin
[2]IOT Laboratory, Faculty of Organization and Informatics, University of Zagreb

Abstract

There are multiple ways to conduct research in computer networking. If possible, real networks and systems are the most accurate way through which to experiment. However, in some cases, large-scale experiments are not feasible in real environments because of the costs, time, and manpower required. In such cases, simulation models and simulators provide researchers with appropriate tools to conduct large-scale and complex experiments.

This paper delivers ready-to-use Network Simulation Environment (NSE) for Visible Light Communication (VLC) simulations in vehicular networks. It implements basic vehicle model that allows multi-channel VLC communication. Models are flexible and ready to use to fit any shape and size of vehicles. Furthermore, NSE includes collision detection as well as Line-of-sight checking for communication channel availability. Although, this new NSE can be used (modified) on any ISO/OSI layer its primary purpose is a custom MAC protocol development and large-scale networks with mobility simulations. Implementation verification is successful, and it was based on simulating regular omnidirectional wireless communication based on new VLC model, both mobile and static layouts.

Keywords:Networksimulation,visible lightcommunication,VLC, VLC radio.

1. Introduction

The visible part of the EM spectrum is unlicensed and free, making visible light a very interesting alternative to the radio part of the EM spectrum. In modern vehicles, although they are equipped with state-of-the-art automotive technology, the full potential of present-day computer technology has yet to be introduced. In a vehicular communication setting, VLC is interesting because most of the components needed to enable VLC are already a part of modern vehicles - specifically, an LED, LASER, or any other light-emitting technology that can be controlled using micro controllers and they can be used as a VLC transmitter (Tx). Similarly, a VLC receiver (Rx) component is usually either a photodiode (PD) or a CMOS camera, which can be found in many modern vehicles (e.g., a front camera for lane tracking, a dashboard camera, or a rear camera for parking assistance) The research community working with the VLC network design lack proper large scale VLC network and traffic simulator.

Network simulation environments are widely used by researchers and network systems engineers to simulate computer networks and network topology and to predict network performance. Basagni et al. have made a detailed analysis of different network simulators [1]. They conclude that there exists a significant gap between a simulation and experimental results. Also, it has been identified that most of the simulators lack proper or any implementation of: Carrier sensing, Backoff timing, Radio propagation model - most of them have acircular coverage estimation, and no interference detection

Table I. NSE selection analysis

NSE	Performance rank [3]	Full stack implementation	Open source	Technology
GlomoSim	2	Yes	Yes	tcl/tk
ns2	3	Yes	Yes	tcl/tk
JiST/SWANS	1	Yes	Yes	Java
Parsec	4	Yes	Yes	Haskell

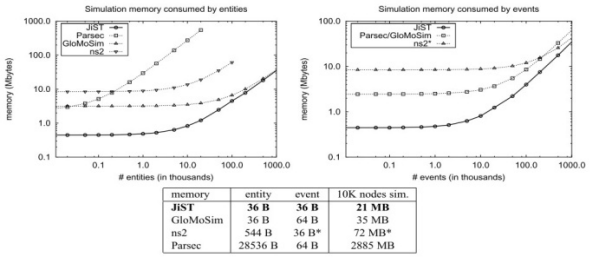
Simulator performance has been recognized as a primary reason for the many gaps identified by authors. Some simulators even lack support for interference detection and have loose PHY model implementation that is a trade-off for increasing simulator performance [2]. None of the widely used simulators: *PhySim, NetSim, OPNET, GloMoSim, cnet, ns2, JiST/SWANS [3], and PARSEC* supports VLC radio communication. For this reason, and because VLC PHY is different from RF, a modular radio model for VLC simulation in a simulation engine must be designed. Many simulators lack obstacle detection (Line of Sight (LOS)) and VLC support in general. VLC technology is not present in network simulators mainly because VLC is a novel technology and VLC applications did not require real network simulations until now. As for current research, *desired simulation engine* for both network and traffic must be selected and modified to support VLC technology.

2. Network Simulation Environment (Nse)Selection

The choice of network simulator follows these guidelines:

- Performance - NSE implementation should be reasonably fast when executed on modern-day environments. It has been identified that performance is the main concern when designing NSE, especially for application layer service simulations (high-end scenarios like video streaming over the Internet).
- Full stack implementation - it is necessary to have full stack implementation to simulate and measure Media Access Control (MAC) performance in real-world scenarios. • Open source - chosen NSE should be *open source* for it to be augmented (enhanced) with new features.
- Technology - the implementation technology used should be Java or .NET (C#), as this is the personal preference of the researcher.

Table I compares different NSE (used in [3]) according to desired guidelines for the NSE selection defined previously.



memory	entity	event	10K nodes sim.
JiST	**36 B**	**36 B**	**21 MB**
GloMoSim	36 B	64 B	35 MB
ns2	544 B	36 B*	72 MB*
Parsec	28536 B	64 B	2885 MB

Fig. 1. Performance analysis of JiST against alternatives [3]

The analysis concluded that: JiST/SWANS NSE satisfies most of the desired guidelines and that it should be considered selected NSE in this research.

3. Jist/Swans Introduction

JiST is an abbreviation of Java in Simulation Time and SWANS stands for Scalable Wireless Ad-hoc Network Simulator. JiSTis a high-performance discrete event simulation engine that runs over a standard Java virtual machine, that is, a Java-based simulation system that executes discrete event simulations both efficiently and transparently by embedding simulation semantics directly into the Java execution model [3]. According to the authors, JiST performance outperforms that of popular network simulators like ns2, Parsec, or GloMoSim. Performance was questionable mainly because of the use of the Java technology against fast and robust C/C++. Figure 1. shows JiST memory performance compared to the alternative simulation engines.

Table II shows CPU performance on a complex scenario with 5 million network events. Metric used is total execution time. Simulations are executed on the same hardware equipment.

Java can be considered a JiST advantage because simulations can be executed on every device, including grid computing networks and server clusters, and even more specialized architectures.

Scalable Wireless Network Simulator (SWANS) is implemented using the JiST architecture. It is a standalone set of classes that utilize the JiST infrastructure to achieve reduced memory demands and high performance. On the ISO/OSI application layer, SWANS can even execute any existing Java network applications using simulated network architecture, thus proving its transparency and real-case-scenario integration in simulations. Unlike its competitors, JiST/SWANS can simulate much larger networks [3]. SWANS can also simulate wired or wireless networks or even wireless sensor networks. SWANS uses a hierarchical binning search/sort mechanism to compute signal propagation in the field; this approach reduces real simulation duration and resources consumption while preserving physical accuracy.

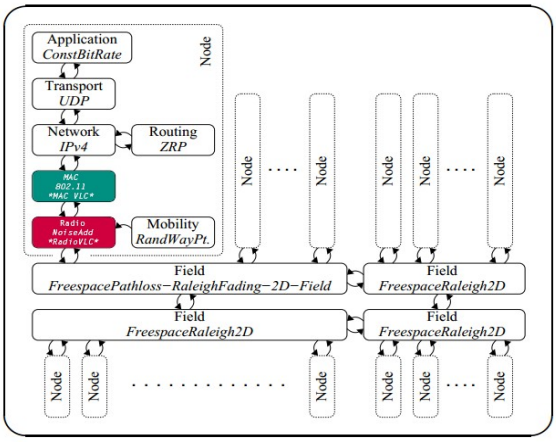


Fig. 2.JiST simulator architecture related to network stack [3]

A. Architecture

JiST/SWANS architecture is a complex system of 550 classes with more than 49000 lines of code. Classes are organized in namespaces, some are shown and described in Table III

B. Features

Existing JiST/SWANS has features related to implementation, such as: simulation time management, extensibility, resources management, and entity invocation. Time management is significant because it introduces three classes of time:

- Actual time: standard Java execution time
- Real time: the real, physical time of execution
- Simulation time: the time inside the simulation; Java instructions have zero duration in the scope of simulation time. Simulation time is advanced using the *sleep* method call.

Network-related features include:

- Full ISO/OSI network stack implementation:
  - Application - heartbeat, CBR, and any Java network application.
  - Transport - UDP, TCP.
  - Network - IPv4.
  - Routing - AODV, ZRP, DSR, GPSR, Geo-routing.
  - Link - 802.11b, loop.
- message - each layer in ISO/OSI has an appropriate message class that follows the standard defined by each layer. Embedding of message content (bytes) is enforced and implemented between ISO/OSI layers.
- field - a concept that includes the physical features of the field (EM) and environment where the network system is simulated.
  - placement - grid, street placement, random, input by XML configuration file.
  - mobility - static, street mobility, random waypoint, random walk, random direction, teleport, XML input file.

Table II. Time to perform 5 million events, normalized against both the baseline and JiST

5 10^6 events	Time (sec)	vs reference	vs JiST
reference	0.738	1.00x	0.76x
JiST	0.970	1.31x	1.00x
Parsec	1.907	2.59x	1.97x
ns2-C	3.260	4.42x	3.36x
GloMoSim	9.539	12.93x	9.84x
ns2-Tcl	76.558	103.81x	78.97x

Table III. JiST/SWANS relevant namespaces and descriptions

Namespace	Description
jist/runtime	JiST simulation framework core
jist/minisim	Regression and benchmark simulations
jist/swans	Wireless ad hoc network simulation components for SWANS
jist/swans/field	Signal propagation, fading, pathloss and mobility, implements EM field
jist/swans/radio	Radio signal transmission and reception, implements radio device e.g. Wi-Fi adapter. PHY layer implementations
jist/swans/mac	Link layer implementations
jist/swans/net	Network layer implementations
jist/swans/route	Routing layer implementations
jist/swans/trans	Transport layer implementations
jist/swans/app	Application layer implementations
jist/swans/app/net	Application sockets
jist/swans/app/io	Application streams
jist/swans/misc	Common data structures and utilities
driver	SWANS simulation configuration drivers

- fading - zero, Raleigh, Rician.
- pathloss - free space, shadowing, two ray.
- field street - complete system for: field, placement, and mobility of nodes in urban environments (streets).
- propagation algorithm - hierarchical binning.
- interference - independent, additive.
- GUI - simple GUI to visualize network systems.

4. Issues

Although JiST/SWANS implements a crucial part of NSE features, there are still features that are missing, caused by the performance issue. Deep analysis of JiST/SWANS code has revealed that it does not even have interference detection. JiST/SWANS has a circular radiation uniform pattern of an RF omnidirectional coverage area, while more accurate would be the spherical estimation. Furthermore, spherical estimation is less accurate than torus [4]. All those shapes define a geometrical, uniform body that should represent the radiation pattern; however, that is also inaccurate because radiation waves do not have a strict border and radiation never ends [5], meaning that a fuzzy torus would be the most accurate approximation of the radiation pattern in simulators.

Sutaria et al. have designed and implemented an energy model for JiST/SWANS [6] because it is not present in the original implementation. The energy model defines the energy consumption of nodes in the network and allows for the comparison of different protocols used based on energy efficiency.

The reason for omitting those features is performance, and JiST/SWANS is not the only NSE that does such a trade-off. Heavy implementation of collision/interference detection on low layers of an ISO/OSI stack would significantly increase resource demands. Deep code analysis has revealed the reason why those features are not implemented; it is the *Link layer*.

In the list of features, it can be noticed that 802.11b is the protocol implemented on the *Link layer*. 802.11 is the standard and well-tested protocol, used in everyday communication using Wi-Fi technology. Because 802.11 works very well on collision avoidance, fully implementing only that protocol would mean that collisions would not occur in the first place and that checking for collisions and interference would not be necessary, also because JiST/SWANS is a network

simulator and not an environment for MAC protocol design. If researchers would like to design and test a new MAC protocol, the existing JiST/SWANS would not be appropriate because one of the core features of a MAC protocol is medium access control, propagation patterns, and collision detection, which must be performed well in a proper MAC protocol. The last feature that is missing in JiST/SWANS is the support for VLC in general. As stated before, JiST/SWANS supports only omnidirectional communication while VLC is mainly directional. Because of that, JiST/SWANS is not capable of simulating networks that use VLC as an underlying technology.

The next section describes missing features implementations and introduces basic network simulator features for the development of a custom MAC protocol.

5. Vlc Enhanced Jist/Swans

Modifying an existing JiST/SWANS simulator to be able to simulate VLC communication requires substantial intervention on several elements (classes) inside the simulator source code. As shown on Figure 2. most modification are done on green and red elements.

To maintain transparency (interoperability), modifications to the Radio class (the red element on Figure 2.) were necessary. Most of the modifications were required to support the testing and design of the MAC protocol.

JiST/SWANS is, among other things, a vehicle network simulator. According to [7], each VLC-enhanced vehicle can have multiple head and taillights; each is able to transmit different data. In practice, this means that each node (vehicle) can have several distinct transmit and receive devices that are, in the scope of this work, called Tx or Rx elements. In Java implementation those are referred to as *VLCelement*class objects.

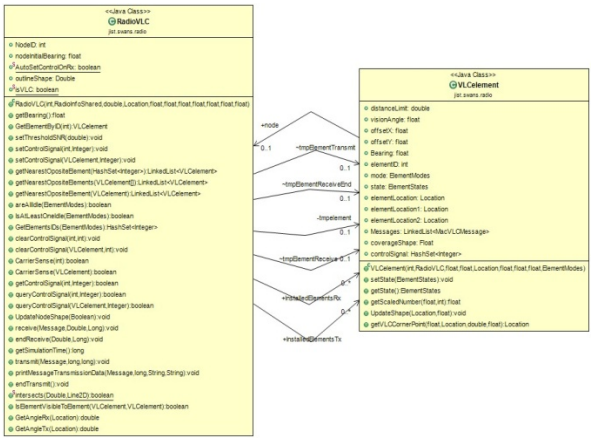


Fig. 3.RadioVLC and VLCelement classes

A. VLC radio

*RadioVLC*is a new class added to the JiST/SWANS to support VLC physical characteristics. Radio type selection is done in the simulation configuration file, described later. In the scope of the simulator, a radio is a transmitting device, e.g., Wi-Fi, and is referred to as a network node, meaning that every network node is a ”radio”. Furthermore, a vehicle capable of communicating is also a network node, meaning that ”radio” is a vehicle in the scope of VLC communication in vehicular networks.

Inherited methods *transmit* and *receive* are the links to the upper and lower layers of the ISO/OSI stack. *Radio* is the implementation of PHY layer characteristics, meaning that the upper layer is Link (MAC) layer. The lower layer, in the context of the ISO/OSI stack, does not exist. However, in JiST/SWANS there is a *Field* class that represents the EM field that implements data transmission and many other features, like fading and pathloss. Information sent by *Radio*, using the *transmit* method, is relayed to the field, and then propagated and delivered to the appropriate nodes - other Radios in the network.

The existing radio uses omnidirectional coverage estimation while VLC has directional coverage estimation. Because of that, it was necessary to track and update the radio bearing (orientation) in space and changes over time, which is necessary for mobility in simulations.

B. VLC element

Figure 3 shows the association between *RadioVLC*and *VLCelement*classes. The VLCelement class implements a Tx or Rx element on a single vehicle. The term “element” is used to define both transmitting and receiving elements. An element is

linked to its radio identified by *R*. The radio class aggregates multiple instances of different elements. Elements are categorized by type; each element can have only one type (mode) of *M*:

- Receive - for receiving the component.
- Transmit - for transmitting the component.
- Unknown - not used.

VLCelement class tracks the state of the element; the state can be:

- Receiving - element is receiving a signal (message) from the field.
- Transmitting - element is transmitting a signal (message) to the field.
- Idle - element is idle and can be used.

It is a violation and impossible for an element of mode *Receive* to be in the state *Transmitting* as well as for an element of mode *Transmit* to be in the state *Receiving*. This rule is enforced on a PHY layer where an exception is raised if such an event occurs.

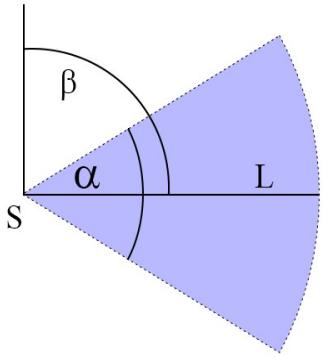


Fig. 4. Element definition (Tx or Rx element)

Element coverage is defined by its physical characteristics.
(Figure 4):

- starting point *S*
- line of sight distance *L*
- vision angle *α*
- bearing angle *β*

The element coverage shape is defined as a circular segment where starting point *S* is a circle centre and *L* is the radius. Bearing *β* is an angle offset where absolute node bearing is a starting angle and equals 0°.

For implementation reasons, each element contains a linked list named *Messages* that contains messages being sent or received at the current moment (in simulation time).

C. MacVLCMessage

As mentioned before, layers (classes) communicate by exchanging messages. A message has a different structure depending on the layer where it is used. For example, a message on the Network layer contains IP addresses, while a message on the Link layer contains MAC addresses, all according to the standards and protocols. Actual message memory size (in bytes) is not equivalent to the number of bytes on a simulated message because *Message* is a Java object, and the network message is a carefully constructed array of bytes according to the standards. *MacVLCMessage*class inherits the structure and implementation of the 802.11 MAC message to maintain compatibility. However, not all elements must be used. For example, this class supports message types:

- RTS - request to send message.
- CTS - clear to send message.
- ACK - acknowledge message.
- DATA - message containing data.

that are used in the 802.11 MAC protocol and that can be used in any MAC implementation.

Table IV.*ElementIDRx HashMap* example:

NodeID	Element1	Element 2	ElementN
0	1	2	
1	2	4	
3	2	6	

A message class designed for VLC was necessary to support the addressing of elements from which a message can be transmitted. To maintain transparency, the same mechanism is used for the receiving element. However, usage and meaning are different: For transmitting, addressing is done using a *HashMap* called *ElementIDTx*, containing the ID of a node and a list of assigned element IDs. Data in this HashMap will instruct the simulator to transmit a message from a specific node using specific elements.

On the other hand, there is a *HashMap* called *ElementIDRx*that contains the ID of a node that has received the message from the medium (field) and the element ID (one or many) that has actually received the message. It is imperative to note that this mechanism is not a future protocol or any related to any standard, and it does not increase the network message size. Actual element

addressing should be done on a protocol level where it will be translated to the mechanism described.

An example of *ElementIDRx HashMap* is shown on Table IV. The first row in the example would mean that the message is received by the node with ID = 0 and it was received on elements with ID 1 and 2. Furthermore, the same message is received by nodes 1 and 3. The assignment of elements to a message is done using the *addElementIDTx(int elementID, int nodeID)* method that uses the node ID and element ID to identify the node and appropriate element. An actual reference for elements and the node is not used in this signature because the caller of this method should be the MAC protocol, while the identification of nodes is done using ID and not the Java object reference. An extra method provided is the one for removing the assignment of node/element; the method signature is *removeElementIDTx(int elementID, int nodeID)*. The described assignment is done for the transmitting element. However, the same principle is done for the receiving element where the PHY layer (VLC enhanced) calls the method *addElementIDRx(int elementID, int nodeID)* that will inform the consumer of the message (MAC) on which the element message was received.

D. Control signal

The control signal is a well-known mechanism that some MAC protocols use to define and detect the channel state. This new feature allows the control signal broadcast that originates from a single element. The receiving element can sense the existence of a control signal in the medium (field). Although it is not a mandatory feature for any MAC protocol, this new feature simply allows for the use of the control signal. It is up to a MAC protocol to manage and respond to the control signal state. In addition, the MAC protocol should limit its performance if the control signal mechanism is used. The reason for actual performance degradation lays in the fact that the actual control signal is created using an exclusively assigned narrow band of the frequency spectrum. The other technique includes segmenting the spectrum in time slots, where one slot is assigned for transmitting the control signal and the rest for the data. If a frequency-based control signal is used, bandwidth should be reduced, and if a time-based control signal is used, message delay should be increased.

Because the control signal transmits only one piece of information (e.g., one bit), existence or not, it is not necessary to check for interference because the control signal is cumulative information on a receiver. The receiver of the control signal does not value the number of control signals set, just its existence. Because of that, interference detection is not necessary for the control signal because interference has the probability to be destructive when there is a concurrent reception of large information, where large means more than one bit.

Control signal implementation was conveniently easier to implement on a PHY layer where, in actual scenarios, the MAC sub-layer would be responsible for control signal implementation. Nevertheless, this feature does not interfere with the validity of network simulation, considering appropriate performance degradation implementation.

The method signature for setting the control signal is: *setControlSignal(int elementID, int channelID)* and it is a member of the *RadioVLC*class (Figure 3). There is also the overloaded method *setControlSignal(VLCelement element, int channelID)*. The first method uses the element ID (integer) to identify the element and is intended primarily for the MAC protocol, whereas the other method uses a reference to the element Java object and is intended for NSE internal use. However, it is public so it can be used by MAC as well. If the element, defined by any of the overloaded methods, is the *Receiving* type, the nearest *Transmitting* element will be used; this is done because the receiving elements cannot transmit any data, including the control signal.

To retrieve the control signal state, the method *getControlSignal(int elementID, int channelID)* can be used. The method returns a Boolean value - *true* if the control signal is present in the medium. Otherwise, it returns *false*. Both *set* and *get* methods have a second parameter called *channelID*which is a number defining the control signal channel. In a simple scenario, a single control signal channelID should be a constant value. This mechanism allows for multiple ”single-bit” control signal channels. It is imperative to note

that in real network scenarios, having multiple control channels would proportionally degrade network performance; as noted before, this performance degradation must be implemented on a MAC level. For frequency band, the work by Triana [8] can be used to calculate performance degradation. His model proved that, for example, the capacity of a channel that is 10 MHz wide is approximately 70 Mbps. A simple reduction of width to 5 MHz gives an approximate channel capacity of 35 MHz. His model proved a linear and proportional correlation between channel width and channel capacity. His work was conducted on a basic IEEE 802.11 MAC protocol.

E. Pathloss

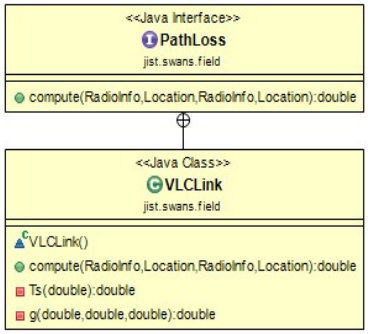


Fig. 5.VLCLink pathloss

Figure5showsthecustompathlossclass(jist.swans.field.PathLoss.VLCLink) that implements VLC pathloss for the visible channel model. Pathloss interface defines one method with the signature *double compute(RadioInfosrcRadio, Location srcLocation, RadioInfodstRadio, Location dstLocation)*. Parameters used define the source radio (node) and its location as well as the destination and its location. Pathloss is calculated between two nodes. The simplification used here is that pathloss is not calculated between individual elements but between nodes itself. The reason for this simplification is to maintain the transparency of existing NSE. The error caused by this simplification is minimal because the location used is the middle point of the node where the element is spaced not more than 1-3 meters, which is reasonable on a close range. On a large range, this difference is insignificant.

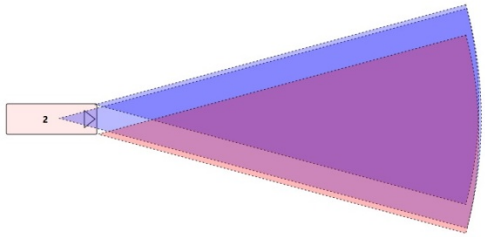


Fig. 6. Element coverage difference

Figure 6 shows the coverage difference between elements with different locations on a typical VLC-equipped vehicle. The difference of position can be viewed as a narrow strip near both edges of the circle segment. The cumulative area of those segments is calculated as:

$S_1 = (0,0)$ $S_2 = (2,-0.75)$ $S_3 = (2,0.75)$

$\alpha_1 = 30°$ $\alpha_2 = 30°$ $\alpha_3 = 30°$

$L_1 = 202m$ $L_2 = 200m$ $L_3 = 200m$

$A_1 = 3400.33\pi m^2$ $A_2 = 3333.33\pi m^2$ $A_3 = 3333.33\pi m^2$

$A_1 - A_2 = 210{,}48 \Rightarrow 1.97\%$

$StripArea \approx 40m^2 \Rightarrow 1.17\%$ (1)

The area difference between shapes 1 and 2 (or 3) equals

1.97 % and double the actual strip area equals $2 * 1.17\% = 2.34\%$ of the actual element coverage area. It can be safely concluded that this simplification is safe, and its performance is more efficient because the computation complexity would increase by the next order of n because each node can have n elements. Thus, the complexity of the *Compute* method remains $O(n^2)$ which is reduced even more by using the hierarchical binning algorithm [9]

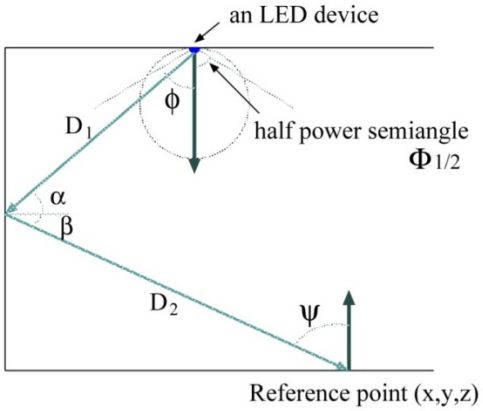


Fig. 7. VLC pathloss model [10]

Implementation of VLC pathloss follows work done by Komine and Nakagawa [10]. The authors demonstrated and defined the model (Figure 7) for

visible light signal propagation in space that uses relative spatial relation between the receiver and transmitter.

F. Collision and interference detection

Interference is a physical condition in which a single receiving component receives two or more concurrent signals. A signal is, in the scope of NSE, one *Message* object. That is why *MacVLCMessage*tracks information on which elements interference occurred in the whole network of nodes. It is up to the MAC protocol to test and respond to the occurrence of interference.

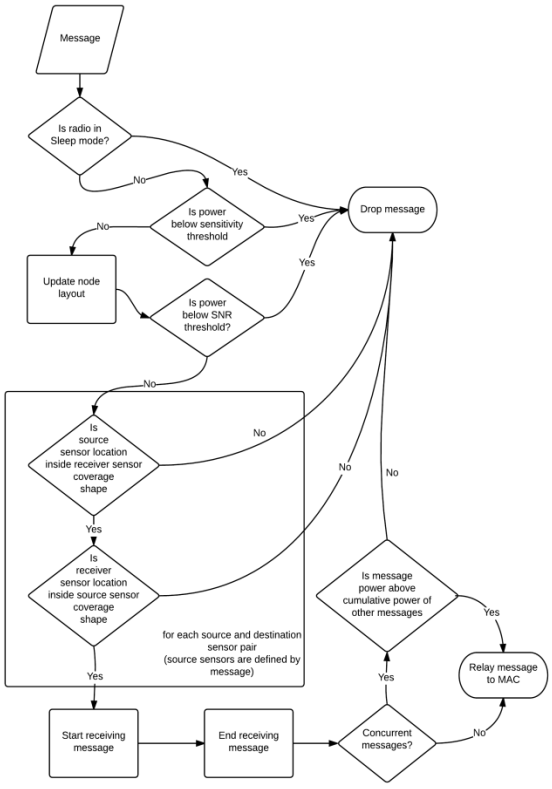


Fig. 8. VLC radio receive message workflow.

An important missing feature is interference detection, especially when designing and testing the MAC protocol. Figure 8 shows a simplified computational workflow of the VLC radio reception of a single message. Interference and the collision detection mechanism are also shown in the figure. JiST/SWANS NSE incorporates two methods: *receive* and *endReceive*. Both have zero simulation time, but the latter is executed after message transmission time. The number of concurrent transmissions is counted in *endReceive*. If the cumulative power level is below the current message power level, the message is successfully received, and the other messages are dropped. This means that interference does not necessarily mean that the message will be dropped. Only if the message power level is below other power levels collision will occur, and the message be dropped. Collision detection considers elements and their coverage areas; because of this modification, it is possible for a single radio to transmit different messages on each installed element. In addition, a single radio can receive multiple concurrent and different data streams.

The use of collision and interference detection code increases simulation computation time compared to no collision detection in the original implementation. However, this was necessary for later stages of research such as MAC protocol performance benchmarks and analysis.

G. LOS detection

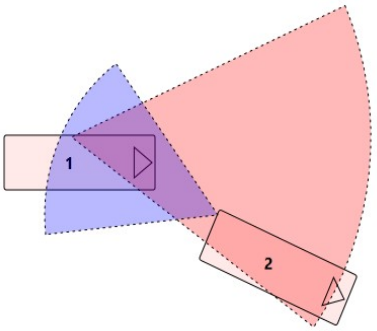


Fig. 9. LOS interruption by vehicle

Figure 8 shows the computational workflow that is optimized to reduce the number of CPU cycles used. Because collision and interference detection consume many CPU cycles for each transmitted message, avoiding that calculation is advisable and is done by enforcing *line of sight communication* only. To detect LOS, each transmitting element should be present in the other vehicle receiving element coverage area. Prior to checking the intersection, a plain distance calculation is used to verify whether vehicles are in the communication range in the first place. After they are in range and coverages areas overlap, intersection detection is enforced. If there is another vehicle between two nodes, LOS is interrupted, and communication is not possible. The LOS interruption detection algorithm, among other vehicles, includes participating vehicles to support unusually placed

elements like the example shown on Figure 9 where the LOS of an element on vehicle 1 towards vehicle 2 is physically interrupted by vehicle 1's outer shape.

If there is a clear LOS, communication is possible and the algorithm advances to collision and interference detection.

H. Configuration

JiST/SWANS uses an XML configuration file where the simulation parameters and configuration are defined. The existing configuration mechanism is very scalable and supports numerous varieties of simulation types. The configuration schema is extended with VLC-specific settings, for example, Figure 10 shows only the relevant fraction of the complete XML file.

Configuration simulation settings that are VLC related are:

- pathloss: an integer value defining the type of pathloss model: *1= Free Space, 2 = Shadowing, 3 = Two Ray, 4= VLC pathloss*.
- placement: an integer value defining the node placement strategy: *1 = Random, 2 = Grid, 3 = Street Random, 4= Street Circuit*.
- StaticPlacementOptions: newly added XML segment that defines static (Grid) placement using a specific structure:in this research is defined as touple:

(R_x,R_y,B,w,l,Δw,Δl,S{...}) (2)

where:

- R_x,$R_y$are coordinates of the centre point of the vehicle (Radio).
- B is vehicle bearing in degrees.
- w is vehicle width in meters.
- l is vehicle length in meters.
- Δw is vehicle uniform (random) width deviation in meters.
- Δl is vehicle uniform (random) length deviation in meters. This and the previous value can be zero if the generated vehicle has static dimensions.
- S {...} is a collection of elements attached (installed) on a vehicle.

The definition of a vehicle can be flexible in terms of dimension (Δw, Δl). If simulation scenarios require large number of different nodes, this feature is convenient where:

where values are defined by the VLC vehicle model defined in Section VI.

Each vehicle can have multiple elements that are defined in the XML configuration. Each element is separated by *S* keyword followed by an element ID (e.g. *S*1). The configuration file can have a multiple number of vehicles that are separated by *;*

- ResultsPath: is a location of the local disk where the simulation MAC performance data file is stored.
- nodes: is the number of nodes in the simulation.
- transmitters: is the number of nodes that are capable of communicating. The number of nodes and transmitters should be less than or equal to the number of nodes defined by StaticPlacementOptions.
- MACProtocol: is a new value that defines the protocol used. In previous versions only MAC 802.11 was used. Possible values are:*MAC 802 11* for the default 802.11 MAC protocol and *MAC VLC V1* for the MAC for VLC prototype.
- MeasurementMode: is a Boolean value defining whether to measure MAC performance data.

```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.4.2_05" class="java.beans.XMLDecoder">
<object class="driver.JistExperiment">
<!-- environment -->
<void property="pathloss"><int>4</int>
</void>
<!-- placement options -->
<void property="placement"><int>2</int>
</void>
<void property="StaticPlacementOptions"><string> 100,100,0,1.7,5,0,0,
s1,22.47,120,0.98667,0.10782,0,t, s2,22.2,60,1,0.06861,0,r; 105.8,102.33,-90,1.7,5,0,0,
s1,22.47,120,0.98667,0.10782,0,t, s2,22.2,60,1,0.06861,0,r;
</string>
```

```
</void>
<void property="ResultsPath">
<string>../results/ra39c5f1f-6ab1-42ea-b3c7-c612847874fc.csv</string>
</void>
<void property="nodes"><int>2</int>
</void>
<void property="transmitters"><int>2</int>
</void>
<void property="MACProtocol"><string>MAC_802_11</string>
</void>
<void property="MeasurementMode"><boolean>true</boolean>
</void>
</object>
</java>
```

Fig. 10. XML configuration file (consolidated)

This was only a fraction of the XML settings options that are available. Some are accessible only when a certain option is used.

Editing of the configuration XML file can be done using the Multiple Concurrent Simulations (MCS) application.

6. Vlc-Equipped Vehicle ForNse

The model of the VLC-equipped vehicle includes only a 2D object because existing JiST/SWANS defines network nodes as single points on a 2D map. 3D would be a more accurate approximation; however, it would require more core changes to the existing NSE and more processing power. As concluded before, approximating transmitter and receiver coverage is sufficient enough to be a single circle segment, and the vehicle is approximated as a rectangle. The model of a vehicle used dimensions). Each vehicle can have multiple elements, although it is common for a vehicle to have two front transmitting elements (headlights) and two rear transmitting elements (taillights). The receiving element is a component that is not commonly present in a modern vehicle; to simplify the model, it can be assumed that receiving elements are attached near the transmitting elements: two in the front and two in the back. If the receiving element is a dashboard camera, the vehicle should have only one receiving element in the middle section of the vehicle front part. Each element is defined as tuple of its characteristics:

$$(S(ID), L, \alpha, S_x, S_y, \beta, M) \quad (3)$$

where:

- $S(ID)$ is an integer defining element ID; it should be a unique number only on a single vehicle. e.g. An element with ID = 1 can be installed on more than one vehicle.
- L is an element line of sight in meters.
- α is a vision angle of the element in degrees.
- S_x, S_y are offset coordinates of an element with a referent point equal to the centre of the vehicle. It is a relative point on a node (vehicle) defined as the float value between -1 and 1 e.g. if S = (0,0) then the element is positioned in the middle of the node (vehicle); if S = (1,1) then the element is positioned at the front left corner of the node. A complete coordinate system for elements on node is shown on Figure 11.
- β is a bearing of the element on a vehicle. It is a relative value where 0°equals vehicle bearing (B).
- M is an element mode. It can be "*t*" for transmitting elements and "*r*" for receiving elements.

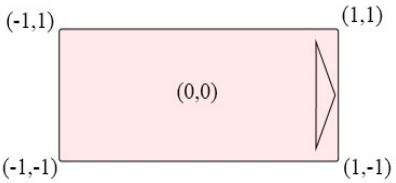


Fig. 11. Node (vehicle) element location coordinates

- The model used in simulations will have two headlight and two taillight elements (Tx), and one PD in the front of the vehicle, and one in the back. According to Wu and Tsai [11], the receiving

element has a vision angle that varies between 60°and 127°.

However, because of PD low cost, the node design of the Rx elements can be flexible to match the cover areas of the Tx elements. Vision length, in the context of the receiving element, is not present because it depends on the incoming light beam, e.g., if the Tx element has enough output power to transmit light that can propagate over 2 km, then PD that is 2 km apart will be able to detect that radiation. According to ECE standard Addendum 111 [12], the vehicle headlight high-beam pattern extends up to 150 m. The shape of the pattern is irregular, two parts shape, the larger shape has a 70°angle.

For rear elements (taillight), ECE standard Addendums 6 and

47 [13], [14] define the element visibility range to be a minimum of 25 m, and the vision angle to be 45°+ 80°where the centre line is perpendicular to the vehicular surface, making it effectually an 80°angle when observing the vehicle as a whole. According to Europe Council Directive 96/53/EC [15], vehicle maximum width is 2.5 m and length is 12 m. Jones created a report regarding vehicle dimensions [16]. He states that the average length of a vehicle is 4.95 m and the average width is 1.52 m. Values vary depending on the vehicle type, though any variation should comply with the EC Directive. The vehicle model used in this research is:

$(0,0,0,1.7,5,0.6,0.8,\{(S1,45,70,1,-1,0,t),$
$(S2,45,70,1,1,0,t),$
$(S3,47,80,1,0,0,r),$
$(S4,45,70,-1,1,180,t),$
$(S5,45,70,-1,-1,180,t),$
$(S6,47,80,-1,0,-180,r)\})$

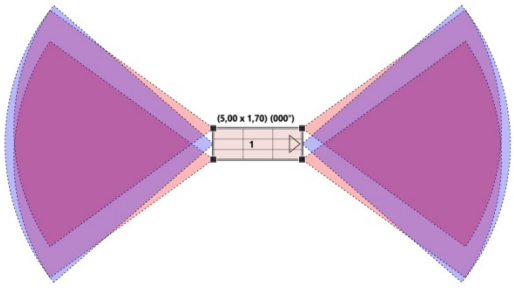


Fig. 12. VLC-equipped vehicle model

A visual representation is shown on Figure 12. This model will be used later in the experiment while evaluating MAC protocol performance. The graphical representation contains only Tx Rx elements with length that is shortened four times. The reason for that is to have a clear and readable printed image. In actual simulations, Tx Rx elements have actual length.

7. Vlc Extended Jist/Swans Validation

The alteration of a large software system requires caution so that those alterations do not interfere with the system's core operations. In the case of JiST/SWANS, it is the network simulation function. Because existing JiST/SWANS software is well tested and verified [17], [18], to validate the new JiST/SWANS version with the VLC modification it would be necessary only to compare network simulation results using the same simulation scenarios before and after modifications. This process is similar to the implementation validation of the MAC protocol for wireless sensor networks in JiST/SWANS done by Tippanagoudar et al. [19].

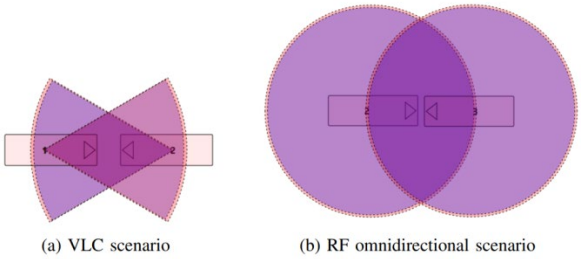


Fig. 13. Validation scenarios for VLC enhanced NSE

Figure 13 shows a graphical representation of a simple scenario that is used to verify integrity, where Figure 13b shows an omnidirectional scenario with only two nodes without obstacles and within the communication range; this scenario is used prior to modifications. Although modifications can support the same scenario, to test all the aspects of the newly implemented features, a VLC radio and VLC relevant scenario are used, as shown on Figure 13a. The VLC scenario uses two nodes (vehicles) where there are no obstacles, and they are within communication range.

For research purposes and validation, the VLC radio can run simulations that use the existing 802.11 protocol with no modifications as a MAC protocol. Because 802.11 does not support VLC element addressing, the VLC radio responds by transmitting the message on every transmitting element installed and receiving concurrently from any receiving element installed.

The validation procedure included 58 simulations for each of two scenarios. Although a single simulation run would be enough to verify integrity, it would be necessary to execute multiple numbers of simulations to reduce random oscillations with simple average function. This randomness is natural in actual network systems and is caused by physical EM spectrum fluctuations and EM random noise in general. This phenomenon is simulated in JiST/SWANS.

Table V. RF and VLC radio comparison on 802.11 MAC

Radio	MAC	Runs	Nodes	Messages	Average PDR (%)
RF Radio	802.11	58	2	628450	88.81412
VLC Radio	802.11	58	2	628450	88.81413

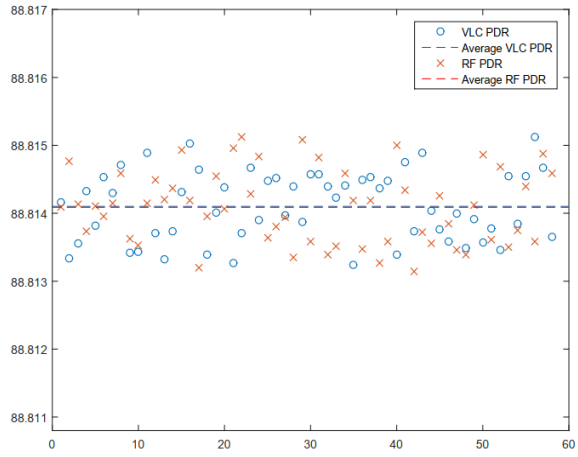


Fig. 14. 802.11 MAC protocol on RF and VLC radio

A summary of simulation results is shown on Table V.

Figure 14 shows a detailed graphical representation of results where the Y-axis shows the average PDR for messages in each simulation run that are represented by the X-axis. On the same plot, average lines are shown for each simulation configuration (VLC and RF). PDR are slightly different on each simulation run regardless of the configuration used; however, the average PDR for every simulation run has a difference of less than 0.00001 %, which strongly confirms the correctness of the implemented alterations. run regardless of the configuration used; however, the average PDR for every simulation run has a difference of less than 0.00001 %, which strongly confirms the correctness of the implemented alterations.

8. Conclusion

MCS is a simple tool that accelerates the process of designing simulation scenarios for JiST/SWANS. This software is open sourced and is under active development. The basic functionality has been implemented and tested on a variety of scenarios and input settings. Testing of MCS was a simple process that included generating multiple configuration files that were loaded in JiST/SWANS. Testing proved that MCS design and JiST/SWANS interpretation of that design were identical in every scenario tested.

MCS software, in the current stage of development, is sufficient for designing simulation scenarios for the MAC protocol evaluation that will be used later.

This section introduced changes to the core of the JiST/SWANS network simulation environment. All changes were oriented towards enabling the design and evaluation of the MAC protocol using new VLC technology. Caution was used when altering core features of JiST/SWANS so as not to interfere with basic functionalities. Nevertheless, software verification was done and proved that the new version does not produce significantly different results from the original version that was already verified, tested, and accepted by the scientific community.

The combination of MCS and JiST/SWANS now supports the design, implementation, and testing of MAC protocols using any technology including VLC. The simple process for designing a new MAC protocol is as follows:

1) Define the new MAC protocol characteristics. 2) Implement the MAC protocol as a Java class in JiST/SWANS.

2) Define the test simulation scenarios and conditions.

3) Design the test scenarios and conditions using MCS software.

4) Start multiple simulations and gather MAC performance results.

5) Interpret the results.

6) The process described above is a guideline for the rest of the research.

Acknowledgment

This work has been fully supported by the Croatian Science Foundation under the project IP-2019-04-4864.

References

[1] C. Vallati, V. Omwando, and P. Mohapatra, "Experimental work versus simulation in the study of mobile ad hoc networks," in Mobile Ad Hoc Networking: Cutting Edge Directions (S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic, eds.), pp. 191–238, John Wiley & Sons, Inc., 2 ed., 2013.

[2] E. B. Hamida, G. Chelius, and J. M. Gorce, "Impact of the Physical Layer Modeling on the Accuracy and Scalability of Wireless Network Simulation," Simulation, vol. 85, pp. 574–588, jun 2009.

[3] R. Barr, Z. J. Haas, and R. V. Renesse, "Scalable wireless ad hoc network simulation," Ad hoc Wireless, and Peer-to-Peer Networks, 2005.

[4] J. Kraus and R. Marhefka, "Antennas for all applications," 2002.

[5] A. Einstein, "The Foundation of the General Theory of Relativity," Annalen der Physik, vol. 354, pp. 769–822, 1916.

[6] T. Sutaria, I. Mahgoub, A. Humos, and A. Badi, "Implementation of an Energy Model for JiST/SWANS Wireless Network Simulator," in Sixth International Conference on Networking (ICN'07), ICN '07, (Washington, DC, USA), pp. 24–24, IEEE, apr 2007.

[7] S.-h. Yu, O. Shih, and H.-M. Tsai, "Smart automotive lighting for vehicle safety," IEEE Communications Magazine, no. December, pp. 50–59, 2013.

[8] J. Tocancipa Triana, "Impact of Channel Width on the Performance of Long-distance 802.11 Wireless Links," Group, 2009.

[9] R. V. Renesse, P. K. Birman, P. A. Mcadams, and R. Barr, "JiST J ava i n S imulation T ime An efficient, unifying approach to simulation using virtual machines motivation : simulation cost per MIPS declining," no. May, 2004.

[10] T. Komine and M. Nakagawa, "Fundamental analysis for visible-light communication system using LED lights," IEEE Transactions on Consumer Electronics, vol. 50, no. 1, pp. 100–107, 2004.

[11] L. C. Wu and H. M. Tsai, "Modeling vehicle-to-vehicle visible light communication link duration with empirical data," 2013 IEEE Globecom Workshops, GC Wkshps 2013, pp. 1103–1109, 2013.

[12] United Nations, "Addendum 111: Regulation No. 112 Revision 3," no. March 1958, 2013.

[13] United Nations, "Addendum 6: Regulation No. 7," no. November, 2012.

[14] United Nations, "AGREEMENT CONCERNING THE ADOPTION OF UNIFORM TECHNICAL PRESCRIPTIONS FOR WHEELED VE-HICLES, EQUIPMENT AND PARTS WHICH CAN BE FITTEDAND/OR BE USED ON WHEELED VEHICLES AND THE CON-DITIONS FOR RECIPROCAL RECOGNITION OF APPROVALS GRANTED ON THE BASIS OF THESE ," no. March 1958, pp. 1–116, 2010.

[15] Ec, "COUNCIL DIRECTIVE 96/53/EC," Official Journal of the European Communities, vol. L 269, no. September 2000, pp. 1–15, 2000. [16] B. Jones, "Vehicles keep inching up and putting on pounds," 2007.

[16] F. Kargl and E. Schoch, "Simulation of MANETs: a qualitative comparison between JiST/SWANS and ns-2," Performance Evaluation, pp. 41– 46, 2007.

[17] E. Weingartner, H. Vom Lehn, and K. Wehrle, "A performance compar-¨ ison of recent network simulators," in IEEE International Conference on Communications, 2009.

[18] V. Tippanagoudar, I. Mahgoub, and A. Badi, "Implementation of the Sensor-MAC Protocol for the JiST/SWANS Simulator," in 2007 IEEE/ACS International Conference on Computer Systems and Applications, pp. 225–232, IEEE, may 2007.