VESNOS User Manual

Ahmed Abdo, PhD

ECE Department, University of California, Riverside

APL, John Hopkins University

Version 0.1

Version History

Version No.	Date	Author	Changes
0.1	April 2024	A. Abdo	Initial version

Contents

1 Introduction to VESNOS

VESNOS, an open-source VANET C++ simulator, facilitates the study of vehicular traffic dynamics, collaborative driving, and vehicle-infrastructure interactions via DSRC-enabled wireless communication and C-v2X. It integrates three main open-source components - SUMO, OMNeT++, and the Security Credential Management System (SCMS)-based V2X simulator - interconnected through the Traffic Control Interface (TraCI).

Supporting multi-modal traffic simulation encompassing motor vehicles, bicycles, and pedestrians, VESNOS enables the design and validation of various DSRC applications. It allows monitoring of message exchange between On-Board Units (OBUs) on vehicles or bikes, Road-Side Units (RSUs), and pedestrians. VESNOS facilitates all types of V2X wireless communications (V2V, V2I, and V2P), enabling the creation of diverse DSRC safety, efficiency, and infotainment applications. Examples include Emergency Electronic Brake Light (EEBL), CACC vehicle platooning, multi-hop routing protocols in VANET, and Traffic Signal Control (TSC) algorithms leveraging real-time traffic information from approaching vehicles.

1.1 VESNOS Architecture

VESNOS is an integrated simulator based on three open-source components (two well-known simulators and a Public Key Infrastructure (PKI)-based approach); one for mobility, one for communication, and the last one for security:

1. **Mobility**: The simulation of traffic in Veins is carried out using the microscopic road traffic simulation package, SUMO (Simulation of Urban MObility) , which supports simulations with or without a graphical user interface and has the capability to import city maps in various file formats. As an open-source, microscopic, continuous space, discrete-time C++ road traffic simulator, SUMO was developed by the Institute of Transportation Systems at the German Aerospace Center and is adopted as our vehicular traffic simulator. VESNOS is closely integrated with SUMO through the Traffic Control Interface (TraCI), harnessing mobility information from cars, bikes, and pedestrians to execute realistic simulations. SUMO enables high-performance simulations of extensive networks with multi-lane roads, encompassing inter-junction traffic, using either simple right-of-way rules or traffic lights. Vehicle types are highly configurable, with each vehicle following either statically assigned routes, dynamically generated routes, or adhering to a configured timetable.
2. **Communication**: is a powerful and versatile open-source network simulation framework designed to facilitate complex computer and communication network modeling and analysis. Written in C++, OMNeT++ allows users to create reusable network modules and define intricate network topologies. OMNeT++ is particularly well-suited for wireless communication simulations for various wireless standards. It incorporates IEEE 802.11p physical layer modeling and IEEE 1609.4 in the Veins framework, facilitating wireless V2X communication among various modules. Additionally, it supports the inclusion of several well-known TCP/IP protocols from the INET framework. Figure 1 illustrates the architecture of VESNOS.
3. **Security**: The SCMS (Security Credential Management System) provides a public key infrastructure mechanism for devices in vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) environments to exchange information in a trustworthy and private manner using digital certificates. Different vehicle makes and models will be able to talk to each other and exchange trusted data by authenticating and validating safety and mobility messages.

1.2 VESNOS Libraries

VESNOS uses the following libraries:

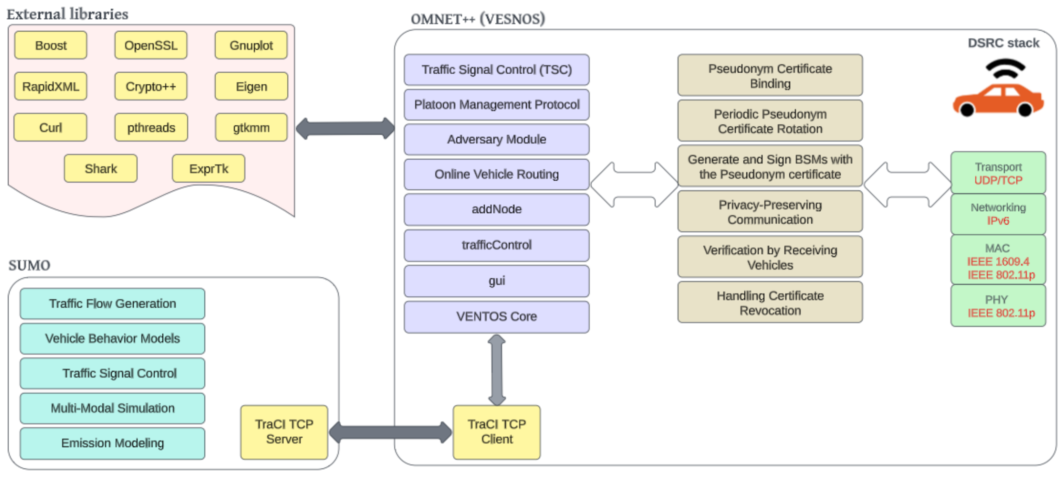

Figure 1: VESNOS system architecture.

1. **Boost**: Boost provides free peer-reviewed portable C++ source libraries. We are using boost filesystem, circular buffer, tokenizer, and graph.
2. **RapidXML**: RapidXml is an attempt to create the fastest XML parser possible, while retaining usability, portability and reasonable W3C compatibility. It is an in-situ parser written in modern C++, with parsing speed approaching that of strlen function executed on the same data. RapidXML is a header only library, and is already included in VESNOS source code.
3. **Curl**: Curl is a library for transferring data with URL syntax, supporting many different networking protocols. We use this library to download the latest SUMO binaries from the Internet.
4. **Eigen**: Eigen is a C++ template library for linear algebra: matrices, vectors, numerical solvers, and related algorithms. We use this library for matrix calculations.
5. **Shark**: Shark is a fast, modular, feature-rich open-source C++ machine learning library. It provides methods for linear and nonlinear optimization, kernel-based learning algorithms, neural networks, and various other machine learning techniques.
6. **Gnuplot**: Gnuplot is a portable command-line driven interactive plotting program.
7. **OpenSSL**: OpenSSL is an open-source implementation of the SSL and TLS protocols. The core library, written in the C programming language, implements the basic cryptographic functions and provides various utility functions. We use OpenSSL library to sign and verify different messages.
8. **gtkmm**: gtkmm is the official C++ interface for the popular GUI library GTK+. We use this library to create the output message log window.
9. **ExprTk**: The C++ Mathematical Expression Toolkit Library (ExprTk) is a simple to use, easy to integrate and extremely efficient run-time mathematical expression parser and evaluation engine. It supports numerous forms of functional, logical and vector processing semantics and is very easily extendible. We use this library to evaluate speed functions.

1.3 VESNOS Installation

We recommend installing VESNOS on the following version of Ubuntu in a dual-boot setup where you will have the most efficient working environment for VESNOS and VENTOS_SUMO. You can use higher versions of Ubuntu but you need to make sure that all the installed libraries are compatible with the Ubuntu version.

Ubuntu	Mac
18 x64	Sierra (10.12)
16 x64	El Capitan (10.11)
	Yosemite (10.10)

You can also use a Virtual Machine (VM) such as Oracle Virtual Box. If you do not use VESNOS often, then the VM would be a good option. This avoids reboots and enables easy data-exchange between the guest and your host OS. Moreover, you can save the VM state using snapshots and roll back to the previous functioning state if something goes wrong. One big disadvantage of using a VM is slow speed because your available computer resources (CPU, RAM, etc.) are shared between two OSes. Hence, if you use VESNOS often, then a dual-boot setup is more suitable. You can install OMNET++ on Windows too, but building VESNOS and VENTOS_SUMO on Windows is very painful and requires several tweaks. Hence, we recommend you to switch to a Unix-like OS such as Ubuntu. Follow these instructions to install VESNOS (and its prerequisites) on your Unix-like OS.

Step 1: If you do not have Git installed on your machine, then install it. [Ubuntu] Run the following command in terminal:

```
sudo apt-get install git
```

Step 2: You need to use the git clone command to download the repository from GitHub using the following command:

```
sudo apt update
sudo apt install git
```

Step 3: You need to download VESNOS repository from GitHub and save it to a folder named (VESNOS_Public) on your computer using the following command:

```
cd Desktop
git clone https://github.com/avmon/VESNOS.git VESNOS_Public
```

If You've downloaded the repository from GitHub,you need to make sure that it is saved under the name VESNOS_Public locally on your computer.

Step 4: You need to download OMNeT++ from the following link:

https://omnetpp.org/download-items/omnetpp/omnetpp-541

Make sure that the downloaded version is compatible with your operating system. Then, move it to the folder (VESNOS_Public). Remember without this step, you get errors installing VESNOS.

Step 5: Go to the (VESNOS Public) folder and run the 'runme' script as shown below. This bash script checks your system and installs the required packages and libraries. **Do not run the script as sudo/root**.

```
./runme
```

You need patience and a fast Internet. The installation process might take more than an hour! Note that the script might show you many compile-time warning messages and you can safely ignore many of them. At

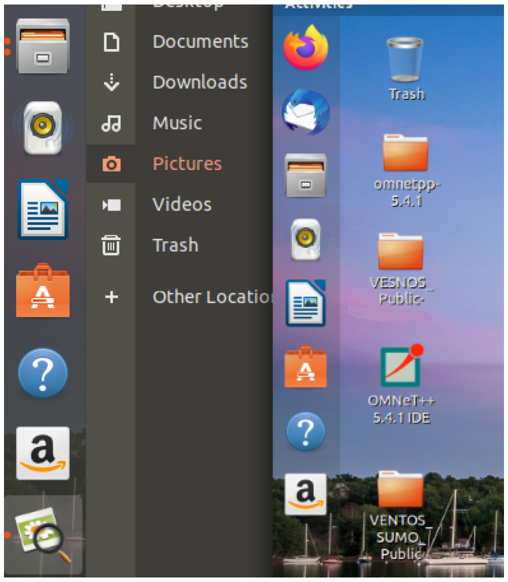

Figure 2: All folders used by VESNOS.

the end you would have three folders (VESNOS_Public, omnetpp-5.4.1 and VENTOS_SUMO_Public), plus OMNeT++ launcher icon on your desktop as shown in Figure 2.

Step 6: Open the OMNeT++ desktop launcher using the following commands:

```
cd ~/Desktop
gedit ./opensim-ide.desktop
```

Replace the line starting with 'Exec' as shown below:

```
Exec=/home/aabdo/Desktop/omnetpp-5.4.1/bin/omnetpp
```

with the following (change the path to omnetpp accordingly):

```
Exec=bash -i -c '/home/aabdo/Desktop/omnetpp-5.4.1/bin/omnetpp;$SHELL'
```

Make sure to aabdo username to your username. Then, save the file and then close it. Now double-clicking the OMNET++ desktop shortcut opens the Eclipse IDE in an interactive shell which loads .bashrc with all the environment variables defined in previous step. If you get an error related to '– add-modules=ALL-SYSTEM', then open the ide/omnetpp.ini file and remove the last three lines.

Step 7: You need to install the extra libraries needed for the security credentials management system as shown in section (1.4) below.

Step 8: You can run the Eclipse IDE using the desktop shortcut or typing 'omnetpp' in terminal. The first time you run OMNET++, Eclipse IDE asks you to select a workspace. Select the folder that you will use to store all your project files. If you have intention to store all your projects on Desktop, then change the workspace to Desktop. Also check "Use this as the default and do not ask again". "Introduction to OMNET++" page will appear. Click on "Workbench". Then it asks you to install INET framework or import some examples into your workspace. Uncheck them both since we do not need them for the time being.

Step 9: Import VESNOS project into the OMNET++ IDE by choosing "File-¿Import" from the menu. Choose "General-¿Existing Projects into Workspace" from the upcoming dialog and proceed with "Next". Choose

”Select root directory” and select the VESNOS folder. ”VESNOS” should appear in the ”Projects” section. Unselect ”Copy project into workspace” if the VESNOS folder is already in your workspace. Click ”Finish”. Step 7: Now you can build the VESNOS project. Use Ctrl+B or right-click on the project name and choose ”Build Project”. Wait for a while and then check the console windows at the bottom of the Eclipse IDE to make sure no errors occurred.

Step 10: To make sure that everything is installed correctly, run a sample simulation scenario. Click Run in the menu bar and select ‘Run Configurations. . . ’. Choose ’OMNET++ Simulation’ and click on ’New launch configuration’ button at the top left. Give this configuration a name like myConfig. In ‘Executable’, choose opp_run and set ‘Working dir’ to /VESNOS/examples/platoon_cacc. In ’Ini file(s)’ choose omnetpp.ini. From ’Config name’ choose a configuration from the drop down list like ‘CACCVehicleStream1’. Leave the rest of the options to default. Click Apply and then click Run.

Step 11: OMNET++ Qtenv graphical runtime environment and SUMO-GUI window appear. Click on the red play button on the toolbar to start the simulation.

1.4 The security credentials management system Libraries

These libraries are required to run the security credentials management system. You need to ensure that each one is installed individually according to the bellow specifications:

1.4.1 OpenSSL (libssl and libcrypto)

OpenSSL is a widely used cryptographic library that provides support for Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols, as well as cryptographic functions.

```
# Ubuntu/Debian
sudo apt-get install libssl-dev

# CentOS/RHEL
sudo yum install openssl-devel
```

1.4.2 Boost Random (libboost_random)

Boost Random is a library that provides facilities for generating random numbers and distributions.

```
# Ubuntu/Debian
sudo apt-get install libboost-random-dev

# CentOS/RHEL
sudo yum install epel-release
sudo yum install boost-random
```

1.4.3 Crypto++ (libcryptopp)

Crypto++ is a C++ cryptographic library that offers a wide range of cryptographic algorithms and functions.

```
# Ubuntu/Debian
sudo apt-get install libcrypto++-dev

# CentOS/RHEL
# Not available in default repositories
```

1.4.4 POSIX Threads (libpthread)

POSIX Threads (pthreads) is a standard thread library available on Unix-like operating systems.

```
# Already included in glibc
```

1.4.5 Network Services Library (libnsl)

The Network Services Library (NSL) provides functions related to network communication and networking protocols.

```
# Ubuntu/Debian
sudo apt-get install libnsl-dev

# CentOS/RHEL
# Provided by glibc-headers package
```