V2V Simulators and Related Software

Last updated: August 2026

Table of Contents

Introduction to V2V

Vehicle-to-vehicle (V2V) is a form of wireless communication designed, among other uses, to allow vehicles to avoid collisions, especially in scenarios where a driver or onboard sensors (e.g., cameras, LiDAR) would not be able to perceive an imminent collision. In V2V, vehicles communicate directly with one another and share safety-related information that includes speed, position, braking, the direction of travel, and more. The primary benefit of V2V is therefore improving the safety of the roads; while estimates vary, V2V is generally expected to help prevent hundreds of thousands of vehicle collisions every year and consequently prevent thousands of roadway deaths. However, for safety and security reasons, it is essential to be able to test and simulate V2V protocols before widespread deployments get underway. There are many V2V simulators, each with its own features. Some of the major ones include:

- **WiLabV2XSim:** Formerly known as *LTEV2VSim*, simulates the resource allocation algorithms in cellular V2X (C-V2X) and the medium contention algorithm in IEEE 802.11p (DSRC/ITS-G5). Simulated C-V2X communication is supported for both LTE- and NR-V2X under either network-controlled (LTE Mode 3, NR Mode 1) or direct (LTE Mode 4, NR Mode 2) Sidelink modes for radio resource allocation. 802.11p communication is also supported using CSMA/CA. Customizable parameters include: vehicle traffic patterns/density, PHY parameters (e.g., modulation and coding) and MAC layer scheduling algorithm (i.e., semi-persistent scheduling) parameters.
- **GEMV^2:** A highly scalable geometry-based channel propagation modeling tool for V2V. The tool is built in MATLAB but GEMV^2 itself is open-source. By importing building outlines and environmental obstacles (e.g., foliage) from open-source mapping tools, GEMV^2 is designed to allow modeling and visualization of V2V signal propagation through any real-world environment. Vehicle mobility is imported from the SUMO traffic simulator.
- **VEINS:** An open-source framework for running network-layer simulations of vehicular networks. VEINS supports V2V in the form of bi-directional communication between a network simulator and a traffic simulator. Users have the option to choose from multiple predefined models included with the simulator and can also choose the specific metrics and logs to track and record in each simulation.
- **VeReMi:** A publicly available family of simulated V2X message datasets designed to evaluate misbehavior detection systems for VANETs. The latest release, VeReMi NextGen (2026), includes labeled message logs, predefined machine-learning splits, and a publicly available extensible dataset generator.
- **PTV Vissim:** This commercial traffic simulator was developed for studying traffic flow and management from the perspective of transportation system engineering. Vissim does not support V2V communication; however, its data can be exported in a format usable by other tools such as GEMV^2. Users can set simulation parameters including such as vehicle speed and density, vehicle type, the number of lanes, traffic control devices, etc.

This technical document introduces a subset of these simulators, includes tutorial information for getting started with them, and finally, provides a summary of their key differences.

WiLabV2XSim (formerly LTEV2VSim)

Introduction to WiLabV2XSim

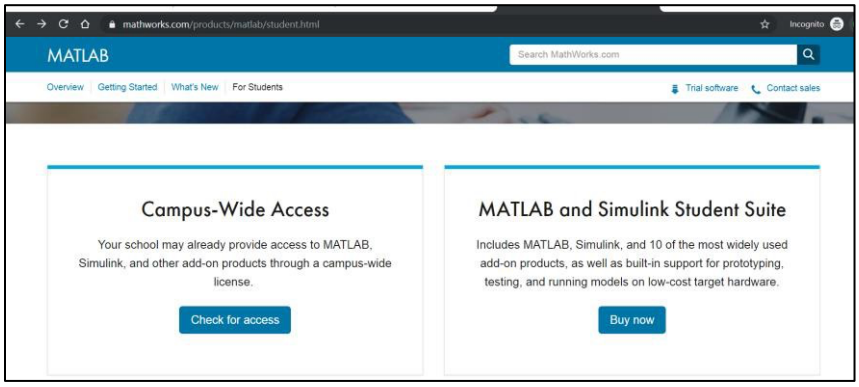
WiLabV2XSim is a dynamic simulator used for the investigation of resource allocation in V2V networks with a focus on cooperative awareness services. It is developed using MATLAB and requires at least MATLAB R2016b for running the simulations. WiLabV2XSim uses vehicle mobility models from 3GPP and ETSI that give realistic vehicle densities, communication ranges, etc. for highway and urban scenarios. Channel models are similarly implemented based on real-world V2V channel measurements from 3GPP, ETSI and 5GAA.

As LTEV2VSim, prior to its 5.0 version (v5.0) it used a "beacon period" timing method for its simulations. This was a compromise between accuracy and running time. In addition, it had two distinct main files for LTE-V2X and DSRC (IEEE 802.11p). Since v5.0, the time granularity has been reduced to 1 ms and there is one single main file instead of two. This allows simulating more cases like traffic generation with non-uniform-periodic characteristics and the single main file allowing the simulation of both technologies simultaneously. From v5.4 (October 2020), there have been multiple improvements for both LTE-V2X and IEEE 802.11p.

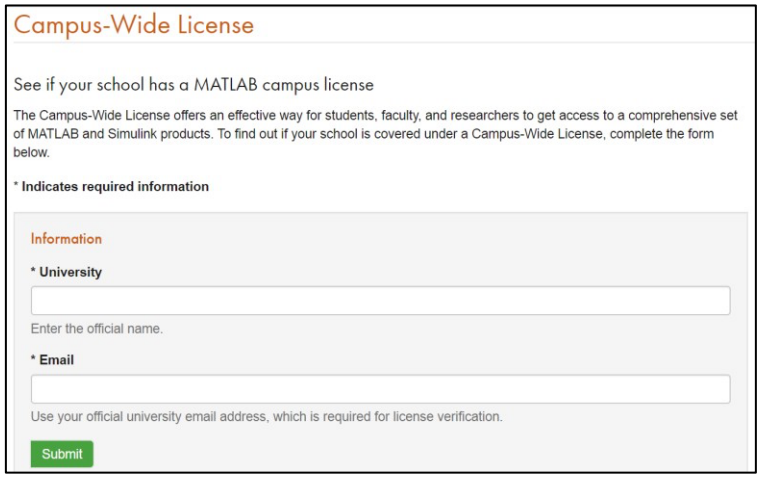
Starting v6.1 (November 2021), it supports sidelink 5G-V2X with focus on the cooperative awareness service. The simulator was consequently rebranded and republished as *WiLabV2Xsim* (https://github.com/V2Xgithub/WiLabV2Xsim).

Installation

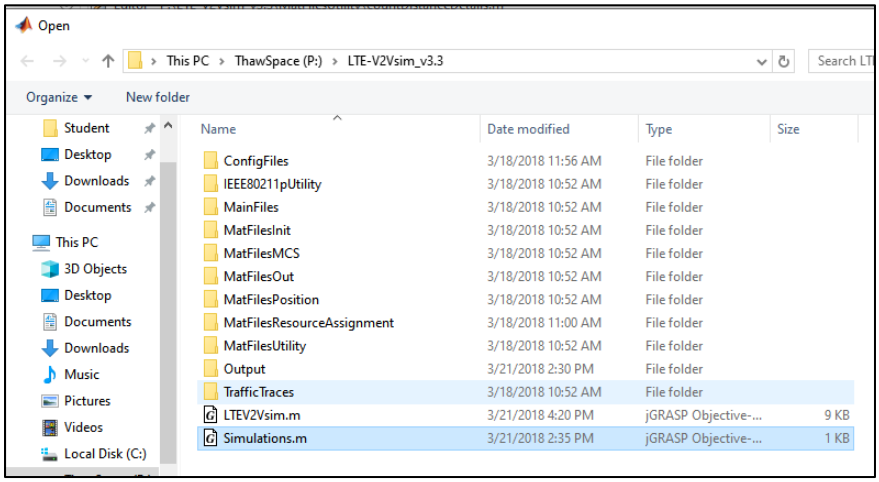
- Follow the below instructions to install MATLAB. If already installed skip to step 6



- Go to https://www.mathworks.com/products/matlab.html. Click on the "For Students" and select the "check for access" under Campus-Wide Access
- Enter the university name and your university email ID.



- You will receive an email in a few moments. Click on the link and create a new account, after which you will be receiving another email to confirm your registration and instructions to download MATLAB
- Download the appropriate file and install MATLAB with default settings and module
- Visit the GitHub Repository https://github.com/V2Xgithub/WiLabV2Xsim
- Download the zip file of the latest version (or any preferred version available) and extract the contents of the file into any preferred location on your machine.
- Open MATLAB and click Open and open the folder in which you have unpacked the simulator files and select the file "Simulations.m"



- Once it opens, click on the 'Run' button
- You will be prompted to change the current folder and click the change folder option

Input parameters and output data

For help regarding input/output parameters, type WiLabV2XSim('help') in the command window.

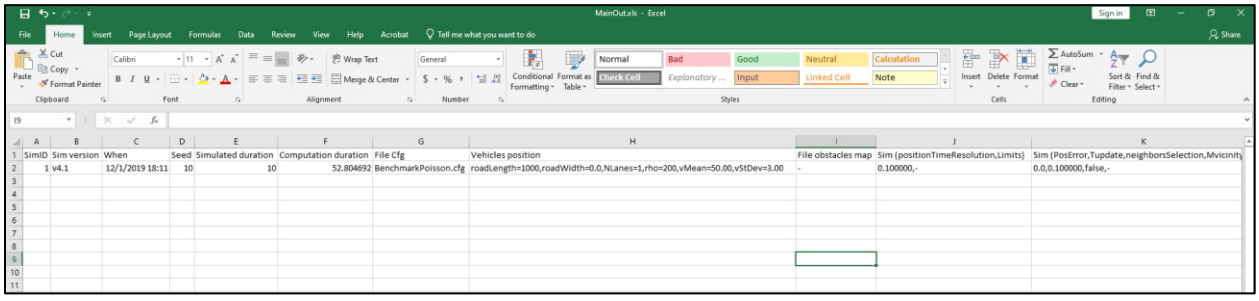
- The "Simulations.m" file contains the initial function call for starting the simulation.
- Once simulation is started, all the initial settings are displayed in the output Window along with their respective variable names and their valid values. You can add any of these variable names to main function call in the "Simulations.m" file.
- The simulation time is also displayed in the output window as below. A brief set of statistics are displayed which are highlighted in yellow.

```
Settings of resource assignement algorithm
Assignment algorithm:   [BRAlgorithm] = 2 (default)
LTE positioning error - 95th percentile (only controlled) (m):  [posError95] = 0.000000 (default)
Time interval between position updates at the eNodeBs (s):  [Tupdate] = 0.100000 (default)
Reuse margin (m):   [Mreuse] = 0 (default)
Interval of scheduled reassignment (BRAlgorithm 2,7,9,10) (s):  [Treassign] = 0.100000 (default)

Simulation Time: 10.0 / 10.0s

Average blocking rate = 0.00000
Average error rate = 0.00170
Packet reception ratio = 0.99830
Average number of neighbors per vehicle = 29.06 +- 5.61
Average number of vehicles in the scenario = 400
```

- To configure simulation settings:
  - Add the variable name within single-quotes followed by its value (Boolean/Number…).
  - Note that Boolean values are case-sensitive, use lower-case
- In this file:
  - The variable *T* has the simulation time. The default value is 10 seconds.
  - The *printNeighbors* is an output setting which is initially set to False. Activating this value create a file and prints the number of neighbours to an excel file in the output folder.
  - The *printUpdateDelay* is also initially set to False. Activating this will create a file and print the update delay between successfully received beacons.
  - The *printPacketDelay* is initially set to false. Activating this will create a file and print the packet delay between successfully received beacons.
- The summary of initial settings after changing the default values are in the image below

```
Application settings
Beacon period (s):  [Tbeacon] = 0.100000 (default)
Beacon size (Bytes):    [beaconSizeBytes] = 300 (command line)
Resource allocated to V2V (%):  [resourcesV2V] = 100 (default)

Physical layer settings
Bandwidth (MHz):    [BwMHz] = 10.000000 (default)
Awareness range (m):    [Raw] = 150 (command line)
Transmitted power (dBm):    [Ptx_dBm] = 23.000000 (default)
Transmitter antenna gain (dB):  [Gt_dB] = 3.000000 (default)
Receiver antenna gain (dB): [Gr_dB] = 3.000000 (default)
Noise figure of the receiver (dB):  [F_dB] = 9.000000 (default)
If using a BLER curve:  [BLERcurve] = false (default)
Modulation and coding scheme:   [MCS] = 3 (default)
Duplexing type: [duplex] = HD (default)
Specify the number of BRs in the frequency domain:  [NumBeaconsFrequency] = -1 (default)
If using adjacent PSCCH and PSSCH:  [ifAdjacent] = true (default)
Subchannel size:    [sizeSubchannel] = -1 (default)
If using Winner+ channel model: [winnerModel] = true (file BenchmarkPoisson.cfg)
Standard deviation of shadowing in LOS (dB):    [stdDevShadowLOS_dB] = 0 (file BenchmarkPoisson.cfg)
Standard deviation of shadowing in NLOS (dB):   [stdDevShadowNLOS_dB] = 4 (default)

Output settings
Folder for the output files:    [outputFolder] = Output (default)
Full path of the output folder = C:\Users\student\Downloads\LTE-V2Vsim_v4.1\LTE-V2Vsim_v4.1/Output
Main output file = C:\Users\student\Downloads\LTE-V2Vsim_v4.1\LTE-V2Vsim_v4.1/Output/MainOut.xls
Simulation ID = 1
Activate the print to file of the number of neighbors:  [printNeighbors] = true (command line)
Activate the print to file of the update delay between received beacons:    [printUpdateDelay] = true (command line)
Enable computation of UD only caused by tx/rx on the same subframe (LTEV2V and HD only):    [enableUpdateDelayHD] = false (default)
Activate the print to file of the wireless blind spot probability:  [printWirelessBlindSpotProb] = false (default)
Activate the print to file of the packet delay between received beacons:    [printPacketDelay] = true (command line)
Delay resolution (s):   [delayResolution] = 0.001000 (default)
Activate the print to file of the details for distances from 0 up to the maximum awareness range:   [printDistanceDetails] = true (file BenchmarkPoisson.cfg)
Activate the creation and print of a PRR map (only for urban scenarios):    [printPRRmap] = false (file BenchmarkPoisson.cfg)
Activate the print to file of the power control allocation: [printPowerControl] = false (default)
Activate the print to file of the hidden node probability:  [printHiddenNodeProb] = false (default)

Settings of resource assignement algorithm
Assignment algorithm:   [BRAlgorithm] = 18 (command line)
LTE positioning error - 95th percentile (only controlled) (m):  [posError95] = 0.000000 (default)
Time interval between position updates at the eNodeBs (s):  [Tupdate] = 0.100000 (default)
Probability to keep the previously selected BR: [probResKeep] = 0.800000 (default)
Percentage of resources to be considered for random selection:  [ratioSelectedMode4] = 0.200000 (default)
Duration of the sensing period, in seconds: [TsensingPeriod] = 1.000000 (default)
Minimum duration keeping the same allocation:   [minRandValueMode4] = -1 (default)
Maximum duration keeping the same allocation:   [maxRandValueMode4] = -1 (default)
Minimum subframe for the next allocation in Mode 4: [subframeT1Mode4] = 1 (default)
Maximum subframe for the next allocation in Mode 4: [subframeT2Mode4] = 100 (default)
Minimum power threshold to consider a BR as occupied in Mode 4, in dBm: [powerThresholdMode4] = -110.000000 (default)
Minimum SINR for a SCI to be correctly decoded, in dB:  [minSCIsinr] = 0.000000 (default)

Simulation Time: 3.4 / 10.0s, end estimated in 36 seconds
```

- The resultant files are saved to the "Output" folder within the WiLabV2XSim folder.
- The output files are generated for each setting individually. For every simulation all the output files are generated again, except for *MainOut.xls*.
- The *MainOut.xls* serves as a log file that records a set of brief settings such as *simulation_id*, *simulator_version*, *simulation_duration* etc... This file is generated only once and for each simulation, new records are added with incrementing *SimID*.



Latest version and changelogs

The changelog for this program is embedded in the project README at https://github.com/V2Xgithub/WiLabV2Xsim.

Veins: Vehicular Network Simulation Framework

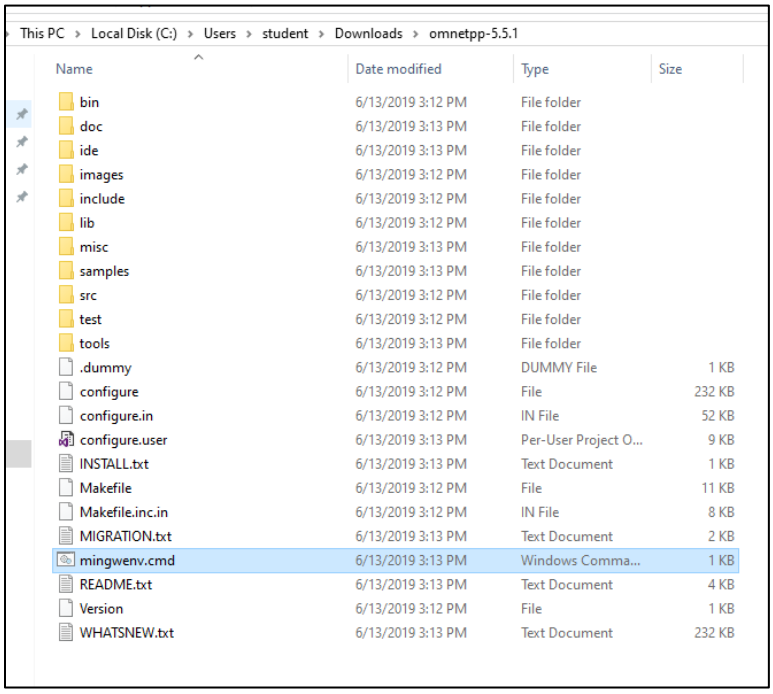
Veins is an open-source framework for vehicular network simulation. The current release, **Veins 5.3.1**, couples the event-based network simulator OMNeT++ with the microscopic road traffic simulator SUMO. The simulators run in parallel and communicate through a TCP connection using SUMO's Traffic Control Interface (TraCI), enabling bidirectionally coupled simulation of vehicle mobility, communication, and traffic behavior. The features of Veins include:

- Vehicle movement generated by SUMO is represented by corresponding mobile nodes in OMNeT++. Communication events can also influence the running traffic simulation, allowing applications such as dynamic rerouting, cooperative driving, and other interactions between vehicular communication and road traffic.
- Veins includes detailed models for vehicular wireless communication, including IEEE 802.11p/DSRC and IEEE 1609.4, together with models for interference, path loss, antenna characteristics, and shadowing caused by buildings and vehicles. Functionality originally derived from **MiXiM** has been integrated directly into Veins, and Veins 5 includes a substantially redesigned and optimized physical-layer implementation.
- The included **Veins_INET** subproject allows Veins mobility to be combined with the INET Framework. This provides IPv4/IPv6 networking and enables integration with additional communication technologies and frameworks, including LTE, C-V2X, and 5G/NR-V2X through projects such as SimuLTE and Simu5G.
- **Plexe** is an actively developed extension for realistic simulation of platooning and cooperative driving systems. The current Plexe 3.2 release supports Veins 5.3.1 and includes realistic vehicle dynamics, cruise-control models, heterogeneous communication technologies, and cooperative maneuvering.
- Veins supports numerous other external extensions, including **space_Veins** for satellite-assisted vehicular networks, **Veins VLC** for vehicular visible light communication, **Artery** for ETSI ITS-G5 protocols, and **PREXT** for evaluating pseudonym-change and VANET privacy mechanisms.
- Machine-learning-oriented extensions include **Veins-Gym**, which provides an interface for reinforcement-learning environments, and **Flexe**, which supports the simulation of federated-learning approaches for connected and autonomous vehicles.
- **Veins_MATLAB** remains an externally hosted example project demonstrating how MATLAB or Simulink models can be called from a Veins simulation. A similar **Veins_Python** project demonstrates integration of Python code with Veins.

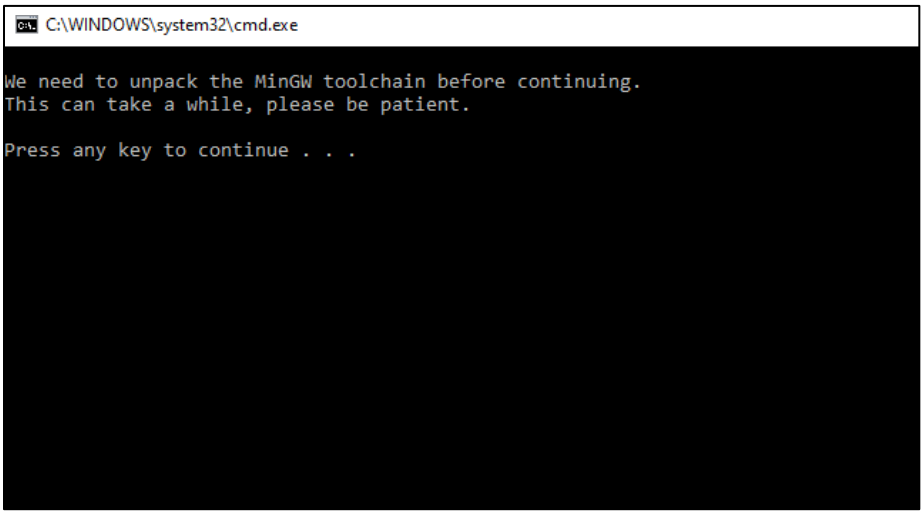
Installation

- Download SUMO from https://sourceforge.net/projects/sumo/files/sumo/
- Install SUMO with the default configuration
- Download the latest version of VEINS from https://veins.car2x.org/download/
- Extract the contents of the zip file to any location of your choice
- Download OMNeT++ from https://omnetpp.org/download/

- Extract the contents of the compressed folder to the same location where VEINS was extracted.
- In the folder of OMNeT++ run the *mingwenv.cmd* file.

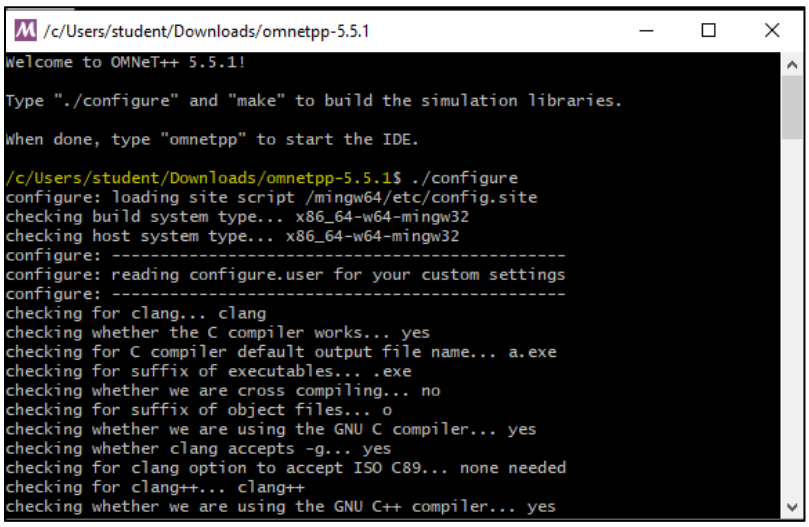


- It will prompt a *cmd* window. Once the command window opens, hit Enter.

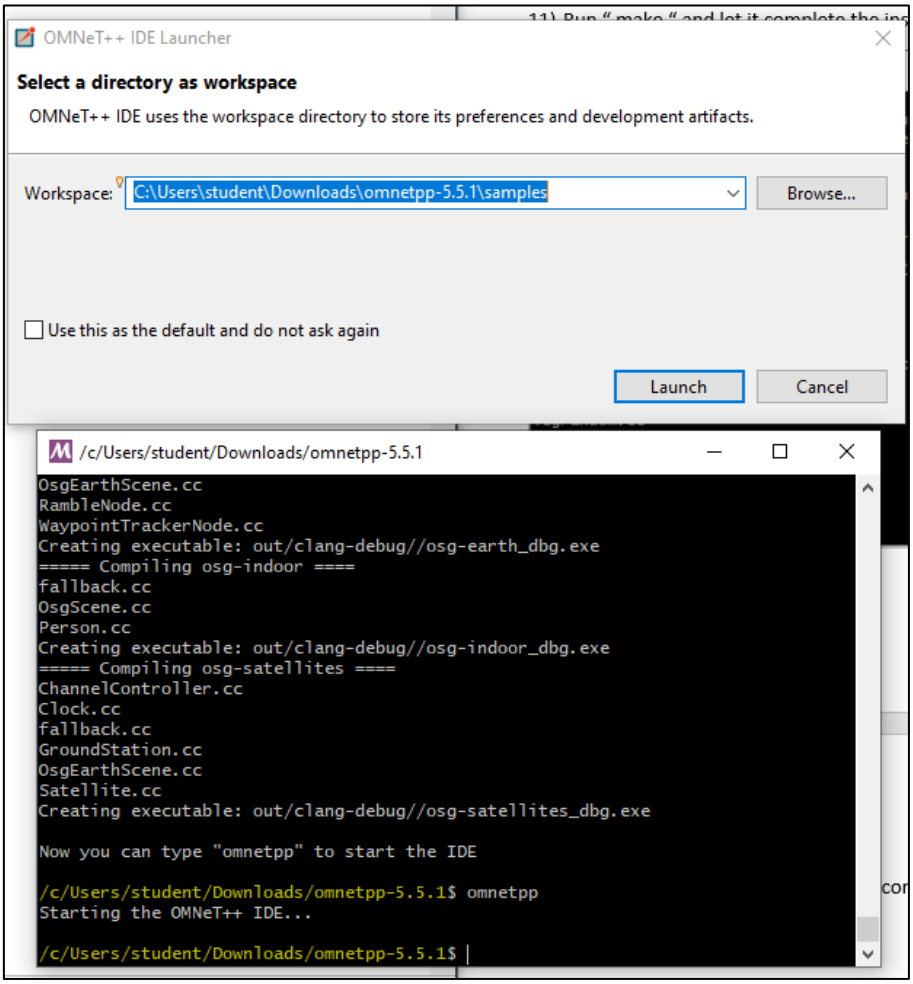


- It will begin extracting the required files. After the extraction of required files is completed, you will receive a prompt. Hit Enter (a couple of times, until a new shell opens).

- Execute *./configure* in the shell and let it complete the configuration.

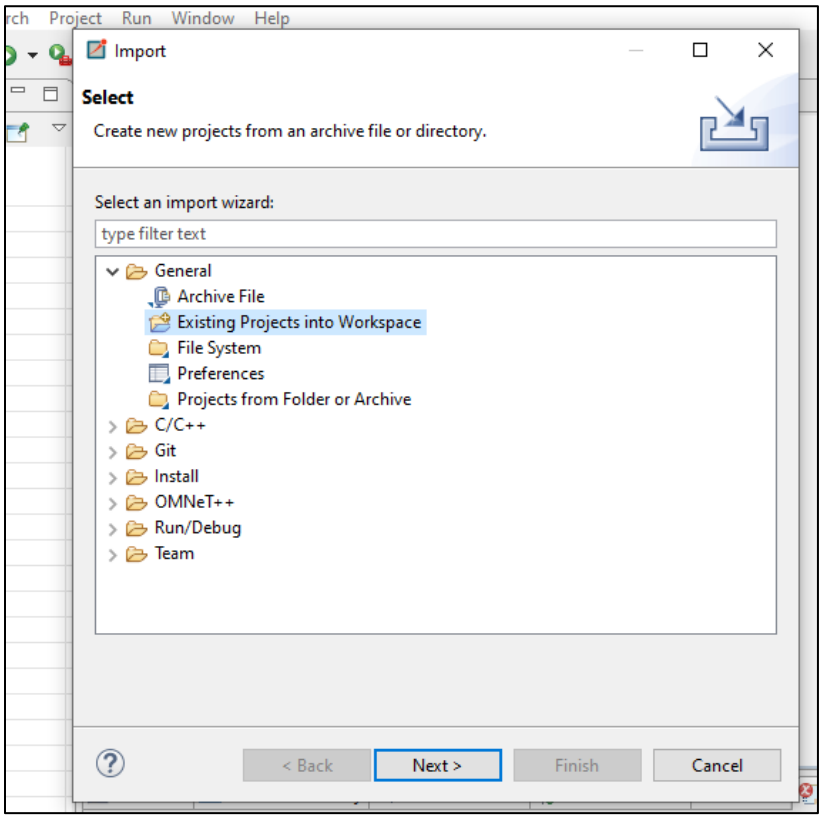


- Execute *make* and let the installation complete (this will take around 20-30 minutes).
- Once it is completed, run *omnetpp* command and a prompt will appear. Click launch in the prompt.

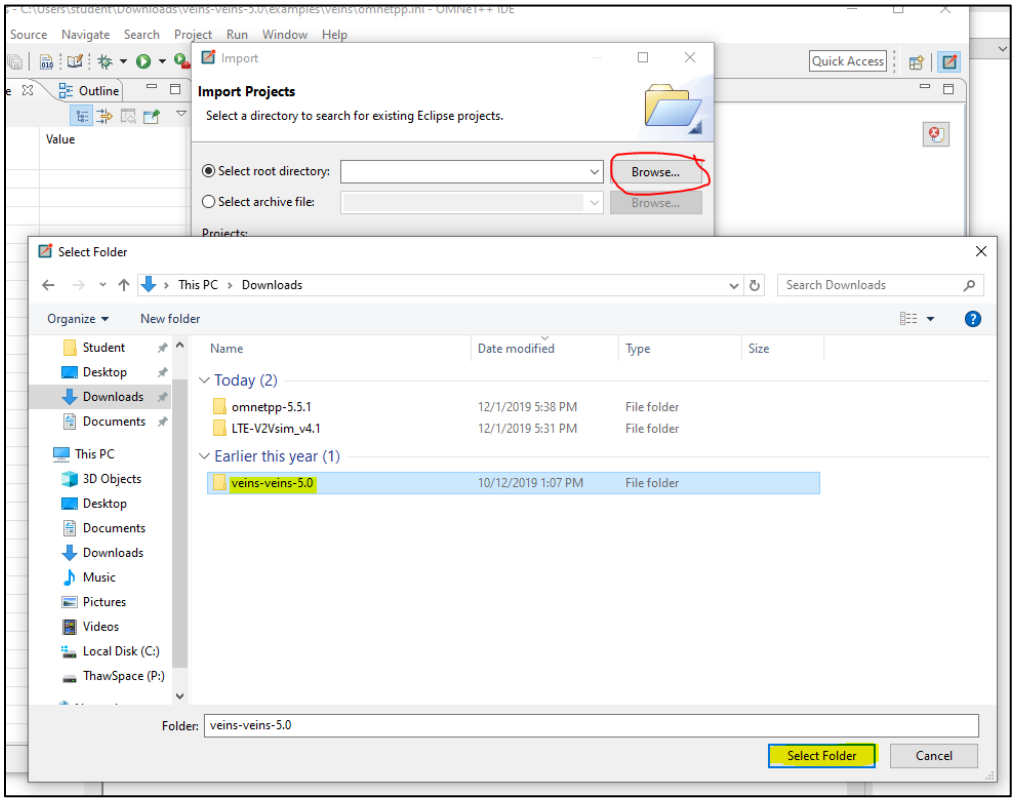


- Now, in the OMNeT++ IDE, click on the workbench icon.

- Open the “File” menu and select “Import”. In the pop-up menu select “General” then select “Existing Projects into Workspace” and click “Next.”

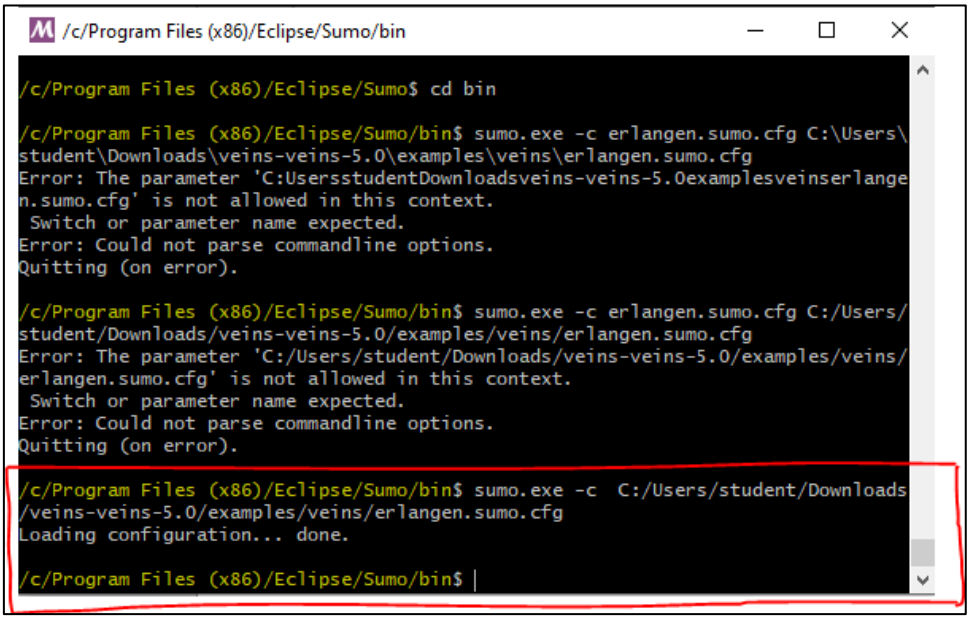


- Select “browse” and navigate to the location where VEINS is saved. Select the folder and click “select folder” and click “Finish.”

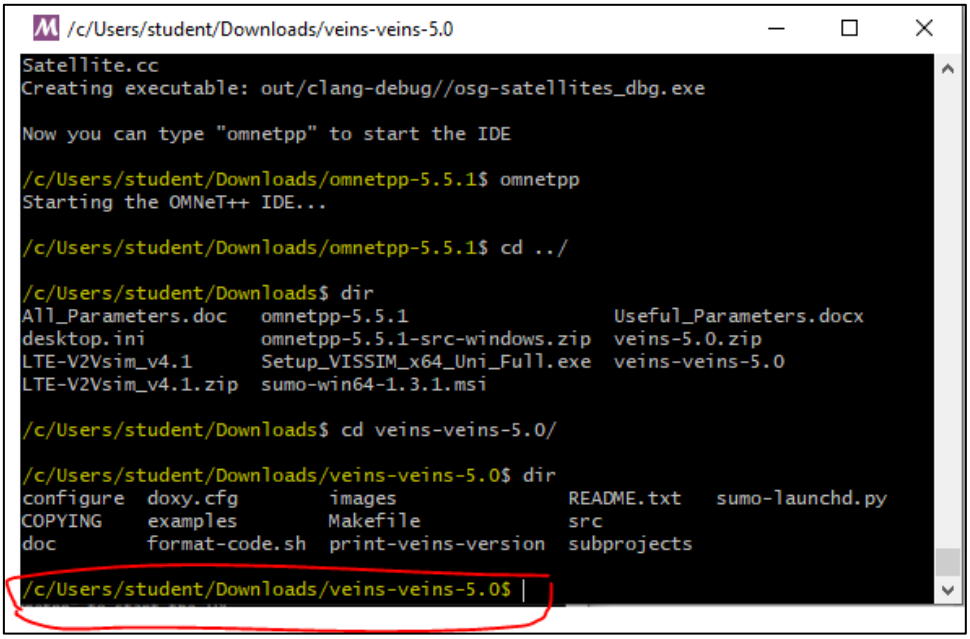


- Select the “Build All” option in the “Project” menu.
- Change the execution directory to *../Sumo/bin* in the *mingw* shell.
- Copy the location where the VEINS folder is located and add the following path to it: *<VEINS_INSTALLATION_DIRECTORY>/examples/veins/erlangen.sumo.cfg*
- Then run the command:

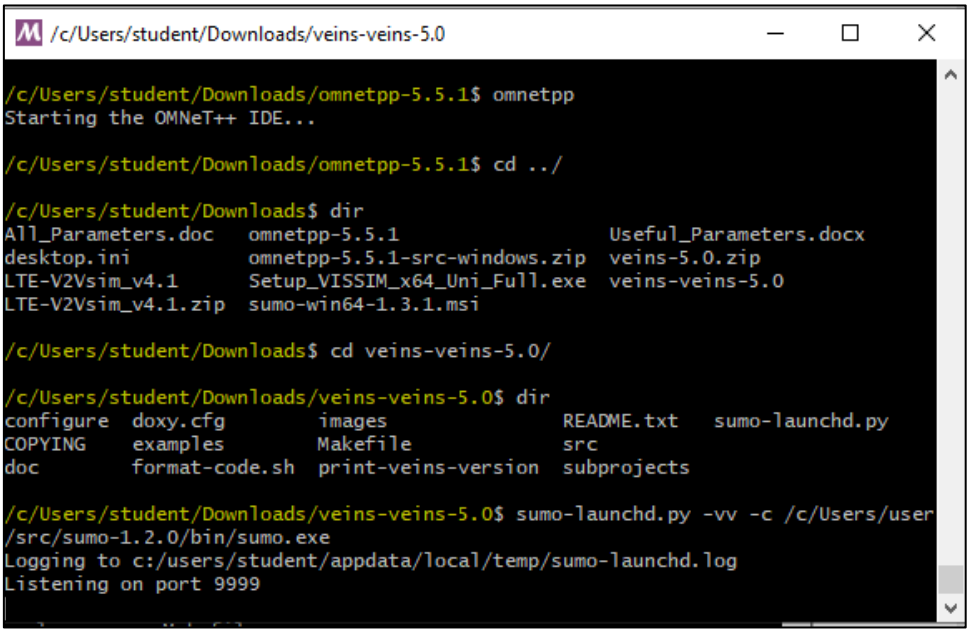
sumo.exe -c <ADD YOUR PATH TO VEINS FOLDER>/examples/veins/erlangen.sumo.cfg



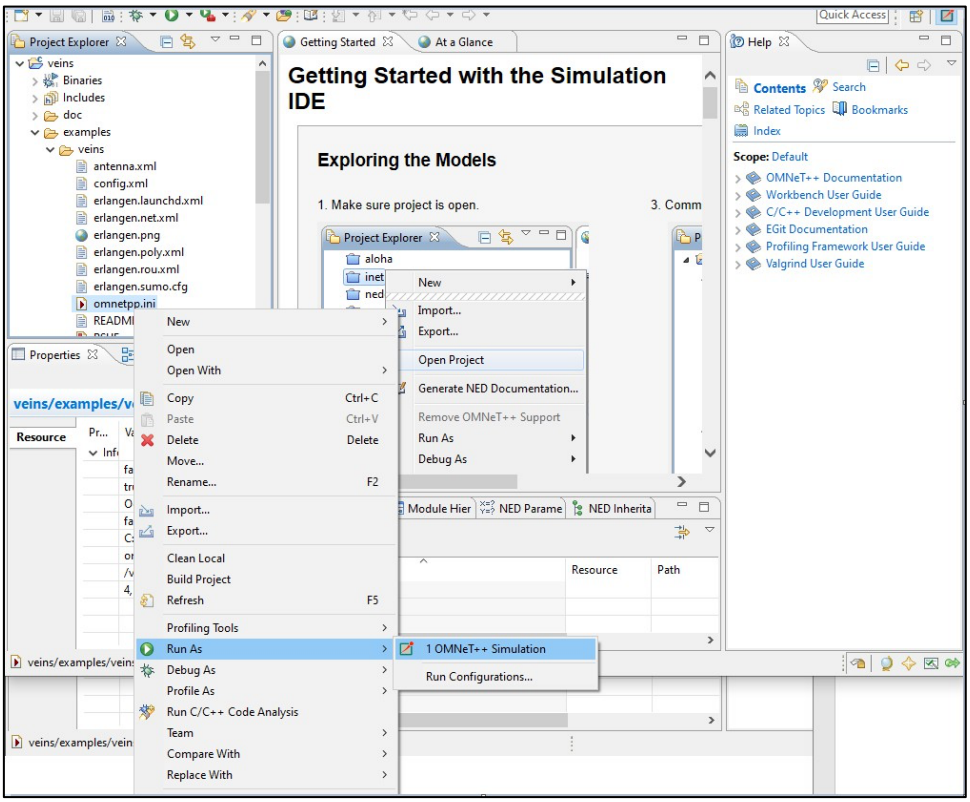
- Change the execution directory to the VEINS folder using *cd <path to VEINS folder>*



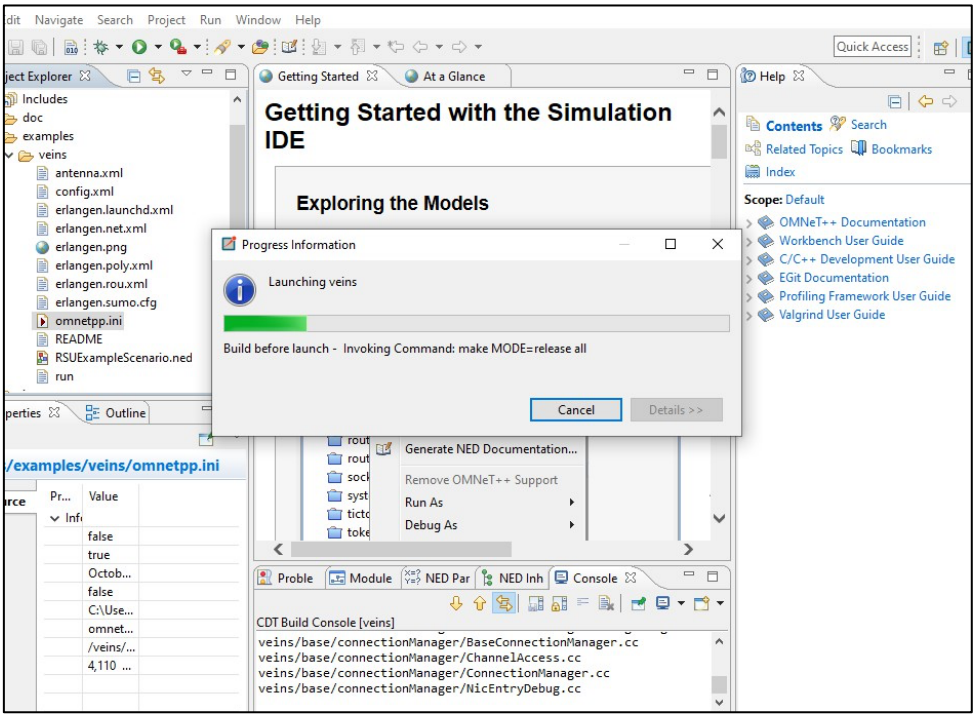
- Then run command *sumo-launchd.py -vv -c /c/Users/user/src/sumo-1.2.0/bin/sumo.exe*



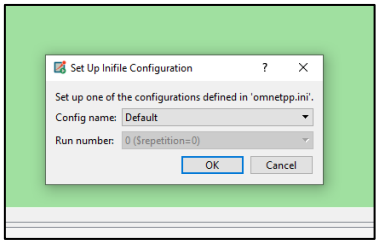
- In the OMNeT++ IDE, go to *examples>veins>omnetpp.ini*. Right click, select "Run As" and then click on "OMNeT++ Simulation."



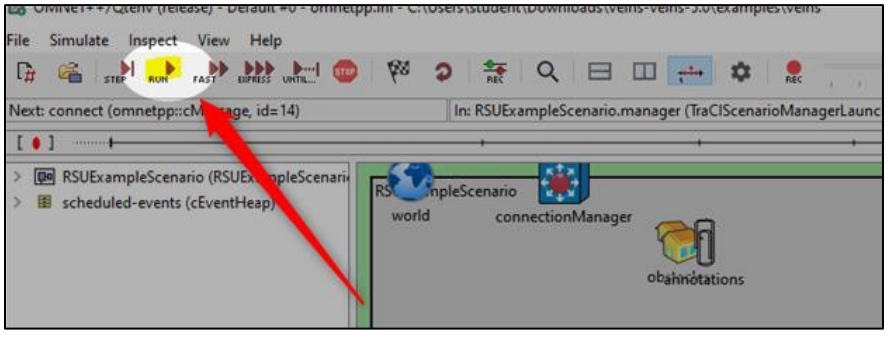
- It will load the simulation.



- After loading we get a prompt, click OK.



- The simulation will be loaded.
- Click the run button to start the simulation.



Latest versions and changelogs

- The latest Veins version is 5.1. The equivalent Instant Veins build is 5.1-i2. From this build of Instant Veins onwards, it will include INET and SimuLTE with every release.
  - The changelogs can be found at https://veins.car2x.org/download/#changelog
  - The GitHub repository for the development version of VEINS can be accessed at https://github.com/sommer/veins/tree/master/
- The latest SUMO version is 1.10.0 which was released on 08/17/2021.
  - The changelogs for this can be found at https://sumo.dlr.de/docs/ChangeLog.html
  - The GitHub repository for the development version of SUMO can be accessed at https://github.com/eclipse/sumo
- The latest OMNeT++ version is 5.7 which was released on 10/06/2021.
  - The changelogs for the latest version 5.7 can be found at https://omnetpp.org/software/2021/10/06/omnet-5-7-released
  - The GitHub repository for OMNeT++ can be accessed at https://github.com/omnetpp/omnetpp

VeReMi: Vehicular Reference Misbehavior Dataset

The purpose of the VeReMi dataset family is to provide a reproducible basis for evaluating and comparing misbehavior detection mechanisms without biasing the evaluation toward a specific detector or traffic condition. **VeReMi NextGen (2026)** is the latest release .

CaTCH-MBD is the reference misbehavior detection implementation provided with VeReMi NextGen. It supports uncertainty-aware plausibility and consistency checks as well as comparison with legacy binary checks. **Maat** was used for evaluation of the original 2018 VeReMi release.

The dataset consists of 180 subsets including:

- 15 different attack types
- 4 scenarios: urban and highway environments with low and high traffic densities
- 3 predefined dataset splits: training, validation, and test
- 3 driver profiles: normal, cautious, and aggressive
- 20% attacker density, with sensor errors and pseudonym changes modeled

The attacks are:

- **Time-related:** Time Delay Attack
- **Position-related:** Constant Position Offset, Random Position Offset, and Position Mirroring
- **Speed-related:** Constant Speed Offset, Random Speed Offset, Zero Speed Report, and Sudden Constant Speed
- **Heading-related:** Reversed Heading
- **Acceleration-related:** Feigned Braking and Acceleration Multiplication
- **Multi-parameter/network attacks:** Sudden Stop, DoS Attack, Traffic Congestion Sybil, and Data Replay

Detection Methods:

- **Range Plausibility:** Checks whether a sender's reported position is consistent with the maximum plausible V2X communication range
- **Position and Speed Plausibility:** Checks whether reported positions are consistent with the road network and whether reported speeds are physically plausible
- **Temporal Consistency:** Compares successive messages to determine whether changes in position, speed, acceleration, and heading are physically consistent
- **Cross-Attribute Consistency:** Checks whether related attributes, such as traveled distance and reported speed, agree with one another
- **Uncertainty-Aware Detection:** CaTCH-MBD incorporates position and speed uncertainty into its checks rather than relying only on binary thresholds
- **Parameter Optimization:** The supplied evaluation tools use automated parameter optimization to tune detector thresholds against the predefined datasets

PTV Vissim

With PTV Vissim, you can demonstrate the impacts of CAVs (Connected Autonomous vehicles) on cities and understand how they interact with regular traffic. You can also experience the simulation in an immersive virtual reality environment and interact freely with other agents.

- Algorithms used for vehicle following and lane change:
  - The model uses the psycho-physical car-following model developed by Wiedemann. It uses vehicle-driver-units that incorporate several stochastic variations. Thus, there are virtually no two vehicles that have the same driving behavior.
  - For lane change, a related rule-based model is used which was originally designed by Sparmann.
- The simulated scenario can be exported from the scenarios list to a standalone file with extension *. Inpx
- Several vehicle and static 3D models are included with the standard Vissim installation. Many additional models in V3D format are available to download through our service area webpage.
- To get accurate results, vehicle movements can also be calibrated in the simulation, so that the driving behavior reflects the local traffic conditions.
- Third-party maps supported by the simulator:
  - The available map providers currently are Bing Maps and OpenStreetMap.
  - Due to licensing restrictions, Google Maps cannot be used as the interactive live map provider in PTV Vissim. You could take a series of screenshots from any map service and add and manually tile them in the Network Editor as Background Images.
- A Poisson distribution is used for the times when a vehicle enters the network. The user can choose whether these arrival times will be determined entirely before simulation start (to allow for the exact number of vehicles).

- Simulating platooning of vehicles is also possible and various attributes related to this feature such as Max. number of platooning vehicles, Max. platoon approach distance, Max. platoon speed etc., can be set by the user. A vehicle can also separate from a platoon.
- Simulate driving errors, including over speeding, lack of attention, distraction and speed misestimation
- The Latest version of PTV Vissim is 2022.00-01 which was released on 11/03/2021. The changelogs can be found at: http://cgi.ptvgroup.com/visionSetups/Setups/VISSIM/220/Release%20Notes_Vissim_2022.00-01_EN.pdf

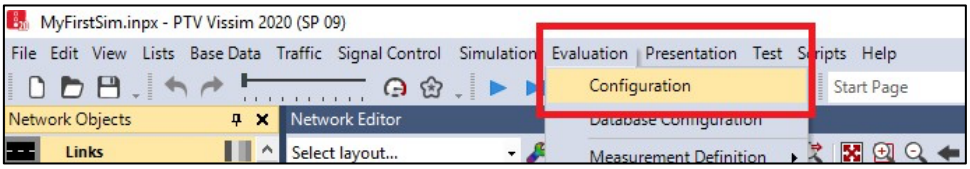
Getting started

Begin with the "First Steps" tutorial. It will walk you through setting up and running a basic intersection management scenario in PTV Vissim, while simultaneously introducing you to the user interface and giving you a basic understanding of how simulations work. A PDF guide for this tutorial, "PTV Vissim - First Steps ENG," can be found in the Tutorials & Guides\Tutorial First Steps directory within your PTV Vissim installation folder. Example scenarios can be in the Examples Demo directory within your PTV Vissim installation folder. There are about twenty example scenarios you can open and run to see how different situations, such as border crossings, roundabouts, or railroad crossings, can be modeled in Vissim.

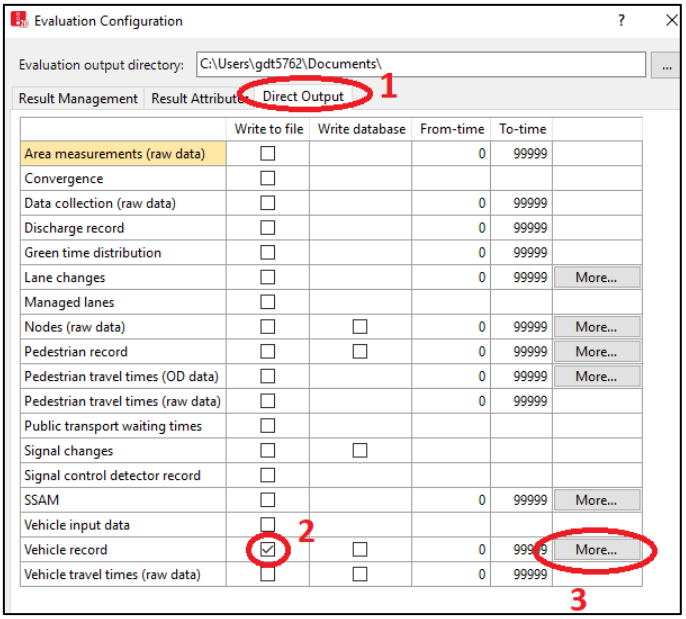
Exporting simulation data

If you want to export vehicle motion data that is recorded during a simulation run, you need to configure Vissim to record data before running the simulation.

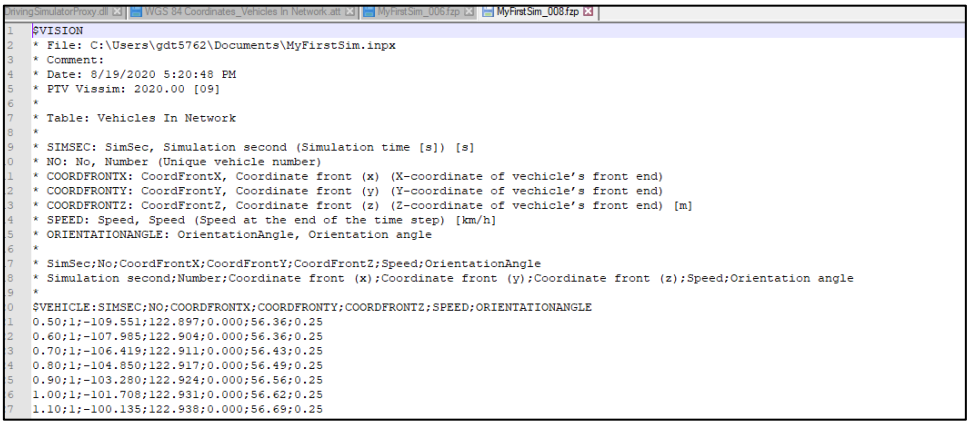
- If you have not opened the simulation file, do that first.
- In the top menu bar, click "Evaluation" and select "Configuration" from the drop-down menu.



- Click the "Direct Output" tab. Next to "Vehicle Record," check the box that says "Write to file." Then, on the same row, click "More"



- Click "Attributes" and delete all of the preexisting selections except for Number.
- From the selection box on the left, add the following data elements:
  - Coordinate front (x)
  - Coordinate front (y)
  - Coordinate front (z)
  - Speed
  - Orientation angle
- Click "OK" on all the open dialog boxes to save the settings. Now, run the simulation.
- After the simulation completes (or is manually stopped), an output file will be saved in your Documents folder ("C:\Users\<your username>\Documents"). The file extension, FZP, will not recognized by Windows, but you can read the file with a text editor like Notepad++:



```
$VISION
* File: C:\Users\gdt5762\Documents\MyFirstSim.inpx
* Comment:
* Date: 8/19/2020 5:20:48 PM
* PTV Vissim: 2020.00 [09]
*
* Table: Vehicles In Network
*
* SIMSEC: SimSec, Simulation second (Simulation time [s]) [s]
* NO: No, Number (Unique vehicle number)
* COORDFRONTX: CoordFrontX, Coordinate front (x) (X-coordinate of vechicle's front end)
* COORDFRONTY: CoordFrontY, Coordinate front (y) (Y-coordinate of vechicle's front end)
* COORDFRONTZ: CoordFrontZ, Coordinate front (z) (Z-coordinate of vechicle's front end) [m]
* SPEED: Speed, Speed (Speed at the end of the time step) [km/h]
* ORIENTATIONANGLE: OrientationAngle, Orientation angle
*
* SimSec;No;CoordFrontX;CoordFrontY;CoordFrontZ;Speed;OrientationAngle
* Simulation second;Number;Coordinate front (x);Coordinate front (y);Coordinate front (z);Speed;Orientation angle
*
$VEHICLE:SIMSEC;NO;COORDFRONTX;COORDFRONTY;COORDFRONTZ;SPEED;ORIENTATIONANGLE
0.50;1;-109.551;122.897;0.000;56.36;0.25
0.60;1;-107.985;122.904;0.000;56.36;0.25
0.70;1;-106.419;122.911;0.000;56.43;0.25
0.80;1;-104.850;122.917;0.000;56.49;0.25
0.90;1;-103.280;122.924;0.000;56.56;0.25
1.00;1;-101.708;122.931;0.000;56.62;0.25
1.10;1;-100.135;122.938;0.000;56.69;0.25
```

NR Module with V2X Extensions (5G-LENA)

The 5G-LENA (ns-3 NR) module is an open-source, pluggable library for the ns-3 discrete-event network simulator. It is developed by Centre Tecnològic de Telecomunicacions de Catalunya (CTTC). To enable comprehensive V2V and V2X testing, the specialized NR V2X extension is distributed as a branch of the core NR module. This extension integrates a customized ns-3 LTE module (managing RLC and upper layers) with the 5G-LENA NR module (handling MAC and PHY sidelink layers) to support standard-compliant vehicular evaluations.

Key features include:

- **3GPP NR Sidelink Support**: Includes support for the NR frame structure, out-of-coverage V2X scenarios, and physical-layer channel modeling compliant with 3GPP TR 38.885.
- **Resource Allocation Mode 2**: Supports autonomous resource selection (Mode 2) using sensing-based semi-persistent scheduling (SPS).
- **PHY/MAC High Fidelity**: Models PSCCH (Physical Sidelink Control Channel) and PSSCH (Physical Sidelink Shared Channel) multiplexing, blind retransmissions, slot-based scheduling, and multi-numerology configurations.

Installation

System Requirements: Ensure a compatible Linux environment (such as Ubuntu or Debian) with essential build tools (g++, make, python3, cmake) installed.

- Install libc6-dev, and sqlite

```
sudo apt-get install libc6-dev
sudo apt-get install sqlite sqlite3 libsqlite3-dev
```

Clone the repo:

```
git clone https://gitlab.com/cttc-lena/ns-3-dev.git
cd ns-3
```

Switch to the ns-3 branch with V2X extensions:

```
cd ns-3-dev
git checkout -b v2x-lte-dev origin/v2x-lte-dev
```

Clone the 5G-LENA NR V2X Module:
Navigate to the contrib/ or src/ directory and pull the specific V2X branch of the NR module:

```
cd src
git clone -b 5g-lena-v2x-v0.1.y https://gitlab.com/cttc-lena/nr.git
```

Configure and Build the Project: Return to the root ns-3 directory and run the configuration script, making sure to enable examples and tests

```
cd ..
./ns3 configure --enable-examples --enable-tests
./ns3 build
```

Run Example Simulation Code

```
$ ./ns3 run nr-v2x-west-to-east-highway
```

```
$  ./ns3 run nr-v2x-west-to-east-highway
Total UEs = 15
Data rate 16000bps
Tx App 1 start time 2.14563 sec
Tx App 1 stop time 12.14563 sec
Tx App duration 10 sec
Tx App 2 start time 2.13551 sec
Tx App 2 stop time 12.13551 sec
Tx App duration 10 sec
Tx App 3 start time 2.10002 sec
Tx App 3 stop time 12.10002 sec
Tx App duration 10 sec
Tx address: 7.0.0.4
Tx address: 7.0.0.9
Tx address: 7.0.0.14
Rx address: 7.0.0.2
Rx address: 7.0.0.3
Rx address: 7.0.0.5
Rx address: 7.0.0.6
Rx address: 7.0.0.7
Rx address: 7.0.0.8
Rx address: 7.0.0.10
Rx address: 7.0.0.11
Rx address: 7.0.0.12
Rx address: 7.0.0.13
Rx address: 7.0.0.15
Rx address: 7.0.0.16
```

Input Parameters and Output

The Input parameters are described as follows:

A. Topology, Mobility, & Node Configuration

- *numVehiclesPerLane* (Default: 5): Represents the number of vehicle UEs generated per lane
- *numLanes* (Default: 3): The total number of parallel lanes running from West to East.
- *interVehicleDist* (Default: 20 meters): The longitudinal distance between the antenna centers

of consecutive vehicles in the same lane.
- *interLaneDist* (Default: 4 meters): The lateral distance separating adjacent lanes.
- *speed* (Default: 38.88889 m/s, approx. 140 km/h): The constant forward velocity of all simulated vehicles along the X-axis.
- *enableOneTxPerLane* (Default: true): When active, only the middle vehicle of each lane is configured as an active packet transmitter.

B. Sidelink (SL) Network & PHY Configuration
- *centralFrequencyBandSl* (Default: 5.89e9 Hz): Defines the central frequency of the Sidelink operational band.
- *bandwidthBandSl* (Default: 400 = 40 MHz): Sets the system bandwidth allocated for the Sidelink.
- *txPower* (Default: 23 dBm): The total transmit power utilized by each vehicle UE .
- *mcs* (Default: 14): The fixed Modulation and Coding Scheme index used for physical-layer transmissions .
- *numerologyBwpSl* (Default: 0): The subcarrier spacing (SCS) configuration for the Sidelink Bandwidth Part (BWP).

C. Resource Allocation & Sensing Settings
- *enableSensing* (Default: true): Enables standard-compliant, sensing-based autonomous resource selection (Mode 2) . If set to false, resource selection occurs without sensing .
- *slSensingWindow* (Default: 100 ms): Defines the duration of the sensing window (T0) used to monitor channel occupancy .
- *slSelectionWindow* (Default: 5): Configures the minimum Sidelink selection window length (T2min) in physical slots, scaled by numerology .
- *slThresPsschRsrp* (Default: -128 dBm): The RSRP threshold used during the sensing procedure to determine whether a resource is busy .
- *slProbResourceKeep* (Default: 0.0): The probability that a UE retains its current resource block allocation after the reservation counter decrements to zero.

After the simulation finishes execution, the engine computes Key Performance Indicators (KPIs). These KPIs are saved back into the database file: *{simTag}-nr-v2x-west-to-east-highway.db*

- **Packet Reception Ratio (PRR):** The proportion of successfully received packets over total transmitted packets, calculated specifically within a defined physical range (default: 200 meters).
- **Packet Inter Reception (PIR) Delay:** The average time elapsed between consecutive packet receptions for vehicles situated within the configured 200-meter range.
- **Throughput:** The aggregate data rate successfully received across all established communication links.
- **Simultaneous PSSCH Tx:** Tracks occurrences where multiple MAC entities attempt to transmit over PSSCH resources concurrently.
- **PSSCH TB Corruption:** Measures Transport Block (TB) error rates and physical-layer packet loss due to interference or poor channel condition.

Comparison of Selected V2V Simulators

	CARLA	WiLabV2XSim	VSimRTI	PTV Vissim	VEINS	ns-3
License	Open Source	Open Source	License given upon request	Commercial	Temporary license given upon request	Open Source
Focus Area	Autonomous Driving	V2V resource allocation and medium contention (PHY/MAC layers)	Connect different simulators in one framework	Traffic control and infrastructure	V2V at the network layer	End-to-end full network stack simulation
Platform	Python	MATLAB	Java	Windows application	OMNeT++ IDE	C++ (Core) / Python (Bindings)
V2V support	No	Yes (802.11p, LTE-V2X, NR-V2X)	Yes (802.11p)	No	Yes (limited)	Yes (802.11p/WAVE, LTE-V2X, NR-V2X)
Customizable aspects	Python code allows all parameters to be edited.	MATLAB code allows all parameters to be edited.	Individual network, traffic simulators can be selected and customized per application.	Highly granular customization of traffic scenarios		C++ classes and APIs allow full protocol customization from PHY to Application layer.
Initial parameters (if any)	Number of vehicles .	PHY settings for chosen technology.	Varies by configuration.	Vehicle speed, vehicle density, number of lanes, direction, signal, speed limit etc.	Specific logs and output to generate must be predefined.	Varies by application configuration
Simulation Logs/results etc.	All events are logged. Messages are shown in command prompt as well.	A single file keeps track of the simulations. Other messages and errors are displayed in the MATLAB console.	Logs are generated for each of the coupled simulators and for VSimRTI. Users can control log level.			PCAP files, traces, custom NS_LOG statements, and performance metrics.

User interaction with the simulator	Different types of vehicles, traffic flow, pedestrians can be simulated.	The simulation is run as program in the background. User has access to the source code through MATLAB.	User can interact with the simulator through a CLI interface			CLI via ns3 runner commands, interacting directly with C++ simulation main files
Models, simulators and algorithms used for simulation	N/A	3GPP highway and urban traffic scenarios are provided for generating messages.	OMNeT++, ns-3, Cell2 and SNS available to support V2V. SUMO is the default traffic simulator. ApplicationSimulatorNT used for simulating V2I infrastructure.			5G-LENA NR module combined with ns-3 LTE module for upper layer execution.
Other supported features	Various vehicle sensors can be simulated and data can be captured for analysis	N/A	Supports other environments and battery simulators for EV. Built-in cellular simulator "Cell2" supports UTMS and LTE.			
Vehicle type selection	Several types of vehicle types are available for selection. Maximum of 250 vehicles per simulation	N/A	Supports the selection of types of vehicles in each simulation through mapping.			Vehicles represented as independent network nodes; mobility is managed via ns-3 MobilityHelper, can also be integrated with sumocfg files

Built-in simulators, extensions and other features	Includes runnable agents, conditional learning agent for autonomous driving. Configuration of LIDAR, cameras, depth sensors, GPS etc. Custom maps can be imported.	Predefined KPIs calculated. Vehicle positions updated by estimations of network or initial trace file Quality assessment of beacon transmission (SINR, PRR).	Simple Network Simulator (SNS) for V2V supports topo-cast (direct uni-or multi-cast communication and geo-cast (communication with clients in a geographical area).	Simulation of driver errors (over speeding, lack of attention, distraction, speed, misestimation) can be configured.	Upper-layer protocols for V2V are supported. MiXM with OMNeT ++ allows modelling transmit power variations.	Supports third-party extension modules such as AI framework, or direct integration with SUMO trace