

V2V Simulators and Related Software

Last updated: Dec. 23, 2021

Table of Contents

1. Introduction to V2V.....	2
2. WiLabV2XSim (formerly LTEV2VSim).....	3
2.1. Introduction to WiLabV2XSim.....	3
2.2. Installation.....	3
2.3. Input parameters and output data	5
2.4. Latest version and changelogs.....	6
3. VEINS	7
3.1. Basic Information.....	7
3.2. Installation.....	7
3.3. Latest versions and changelogs	16
4. Vehicular Reference Misbehavior Dataset (VeReMi).....	16
5. PTV Vissim	17
5.1. Getting started	18
5.2. Exporting simulation data	18
6. Comparison of Selected V2V Simulators.....	20

1. 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.
- **GEMV²:** A highly scalable geometry-based channel propagation modeling tool for V2V. The tool is built in MATLAB but GEMV² itself is open-source. By importing building outlines and environmental obstacles (e.g., foliage) from open-source mapping tools, GEMV² 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 dataset of vehicle messages recorded from VEINS simulations and specifically designed to test designs for misbehavior detection systems that might be used in V2V infrastructures. Simulation logs as well as simulation scripts are available.
- **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 GEMV². 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.

2. WiLabV2XSim (formerly LTEV2VSim)

2.1. Introduction to WiLabV2XSim

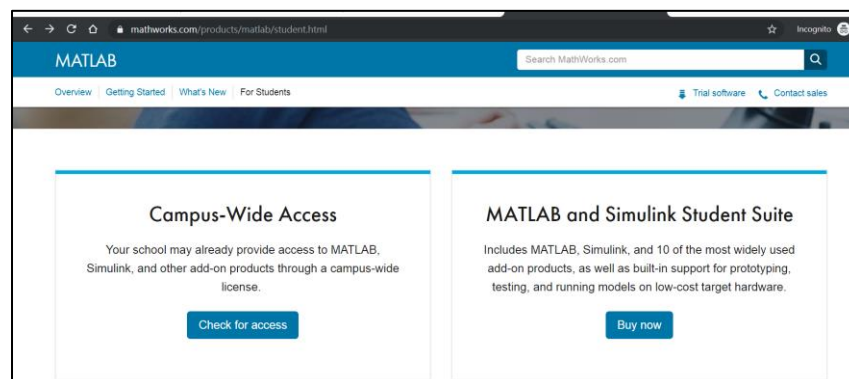
WiLabV2XSim is a dynamic simulator used for the investigation of resource allocation in V2V networks with a focus on cooperative awareness services. It is developed using MATLAB and requires at least MATLAB R2016b for running the simulations. WiLabV2XSim uses vehicle mobility models from 3GPP and ETSI 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>).

2.2. 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.

Campus-Wide License

See if your school has a MATLAB campus license

The Campus-Wide License offers an effective way for students, faculty, and researchers to get access to a comprehensive set of MATLAB and Simulink products. To find out if your school is covered under a Campus-Wide License, complete the form below.

* Indicates required information

Information

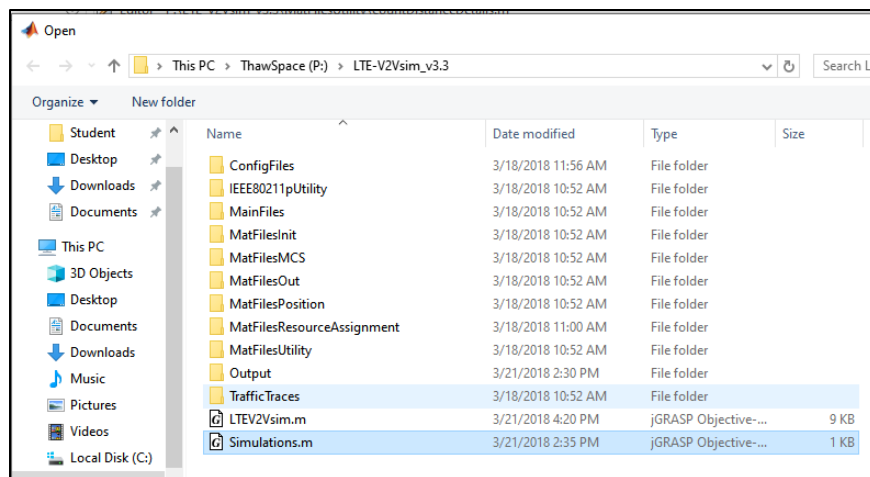
* **University**

Enter the official name.

* **Email**

Use your official university email address, which is required for license verification.

- 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

2.3. 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: [BRAAlgorithm] = 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 (BRAAlgorithm 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 assignment 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*.

SimID	Sim version	When	Seed	Simulated duration	Computation duration	File Cfg	Vehicles position	File obstacles map	Sim (positionTimeResolutionLimits)	Sim (PosError,Tupdate,neighborsSelection,Mvicinity)
1	v4.1	12/1/2019 18:11	10	10	52.804692	BenchmarkPoisson.cfg	roadLength=1000,roadWidth=0.0,NIanes=1,rho=200,vMean=50.00,vstDev=3.00	-	0.100000,-	0.0,0.100000,false,-

2.4. Latest version and changelogs

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

3. VEINS

3.1. Basic Information

Veins is an open-source framework for running vehicular network simulations. It uses two simulators in parallel: An event-based network simulator called OMNeT++ and a road traffic simulator called SUMO. Both simulators are connected via a TCP socket which use a standardized communication protocol interface called Traffic Control Interface (TraCI). This provides a comprehensive suite for the Inter-Vehicular Communication simulations.

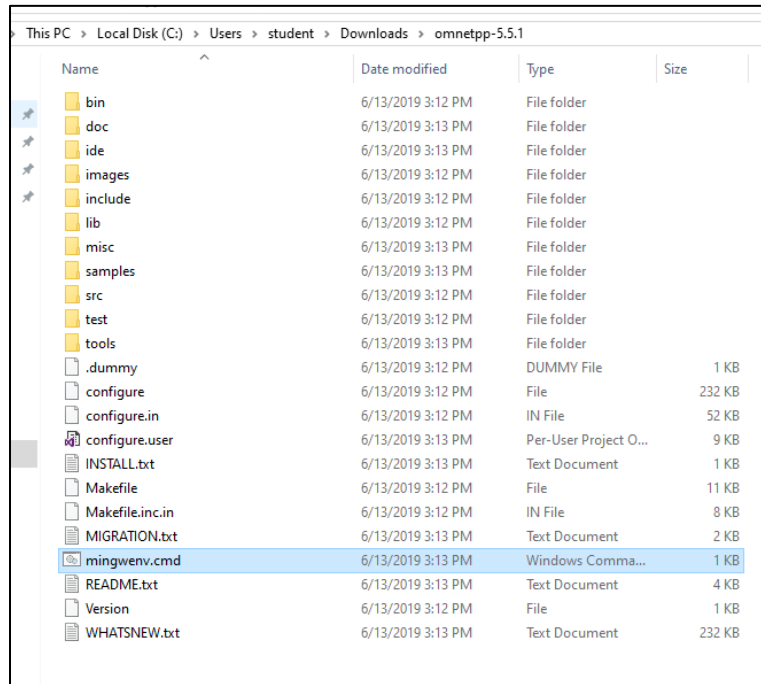
The features of VEINS include:

- Movement of vehicles in the road traffic simulator SUMO is reflected as movement of nodes in the simulation. Nodes interact with the running road traffic simulation to simulate the influence of IVC on road traffic.
- OMNeT++ uses a modelling framework called MiXiM which was created for mobile and fixed wireless networks (wireless sensor networks, body area networks, ad-hoc networks, vehicular networks, etc.). Veins relies on its functionality for accurate modelling of physical layer effects, especially for modelling and working with the distribution of transmit power over time and space.
- In addition to the integrations with other simulators, Veins also offers extensions like Plexe and PREXT.
- Plexe is an extension of Veins which permits the realistic simulation of platooning (i.e., automated car-following) systems. It features realistic vehicle dynamics and several cruise-control models, permitting the analysis of control systems, large-scale and mixed scenario, as well as networking protocols and cooperative manoeuvres.
- PREXT (Privacy Extension for Veins) is a unified and extensible framework that simulate pseudonym change schemes (i.e. privacy schemes) in VANET. It supports seven privacy schemes of different approaches including silent period, context-based and mix-zone and can be easily extended to include more schemes. It includes adversary modules that can eavesdrop vehicle messages and track their movements. This adversary is used in measuring the gained privacy in terms of several popular metrics such as entropy, traceability and pseudonym usage statistics.
- Veins_MATLAB is an externally-hosted project that serves as an example of how to use MATLAB or Simulink models from a Veins simulation.

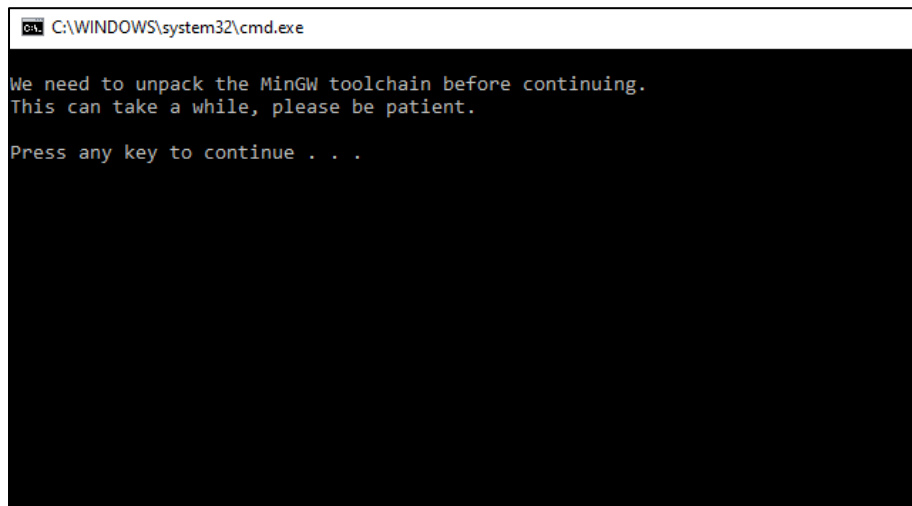
3.2. Installation

- 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/>

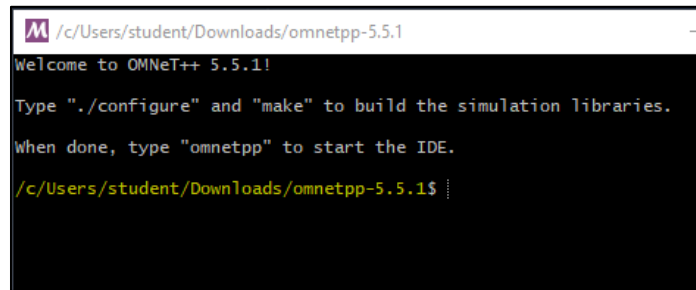
- Extract the contents of the compressed folder to the same location where VEINS was extracted.
- In the folder of OMNeT++ run the *mingwenv.cmd* file.



- 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).



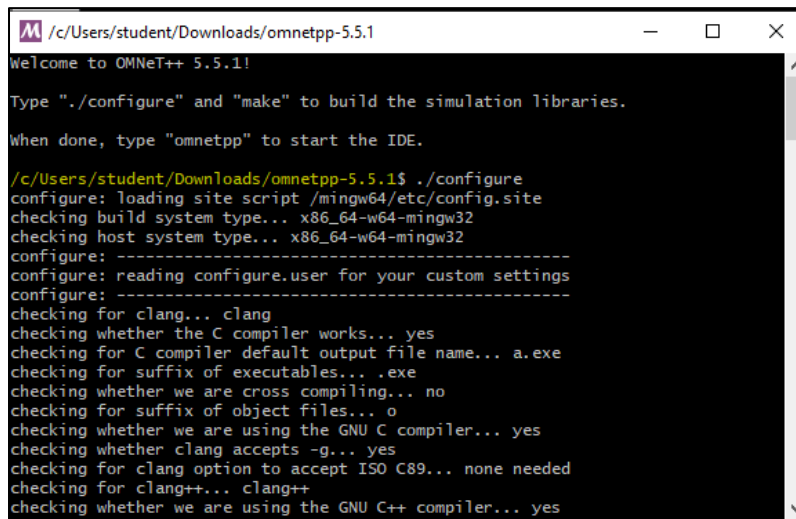
```
/c/Users/student/Downloads/omnetpp-5.5.1
Welcome to OMNeT++ 5.5.1!

Type "./configure" and "make" to build the simulation libraries.

When done, type "omnetpp" to start the IDE.

/c/Users/student/Downloads/omnetpp-5.5.1$
```

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



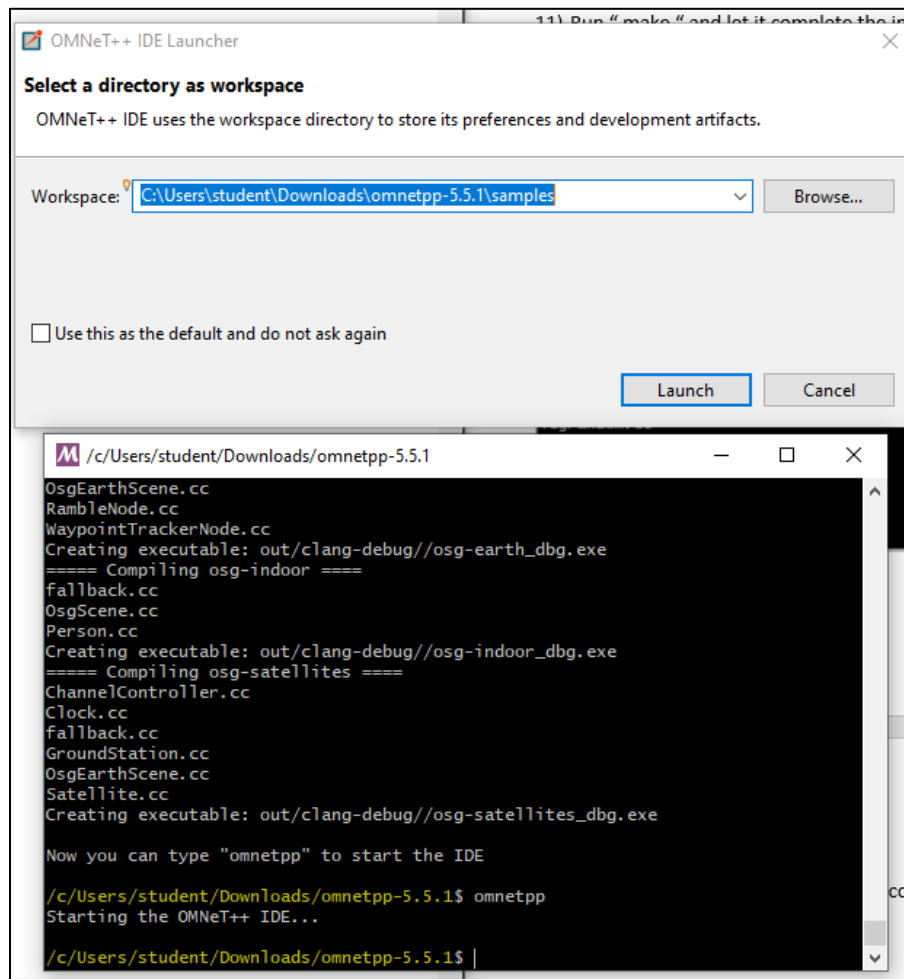
```
/c/Users/student/Downloads/omnetpp-5.5.1
Welcome to OMNeT++ 5.5.1!

Type "./configure" and "make" to build the simulation libraries.

When done, type "omnetpp" to start the IDE.

/c/Users/student/Downloads/omnetpp-5.5.1$ ./configure
configure: loading site script /mingw64/etc/config.site
checking build system type... x86_64-w64-mingw32
checking host system type... x86_64-w64-mingw32
configure: -----
configure: reading configure.user for your custom settings
configure: -----
checking for clang... clang
checking whether the C compiler works... yes
checking for C compiler default output file name... a.exe
checking for suffix of executables... .exe
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether clang accepts -g... yes
checking for clang option to accept ISO C89... none needed
checking for clang++... clang++
checking whether we are using the GNU C++ compiler... yes
```

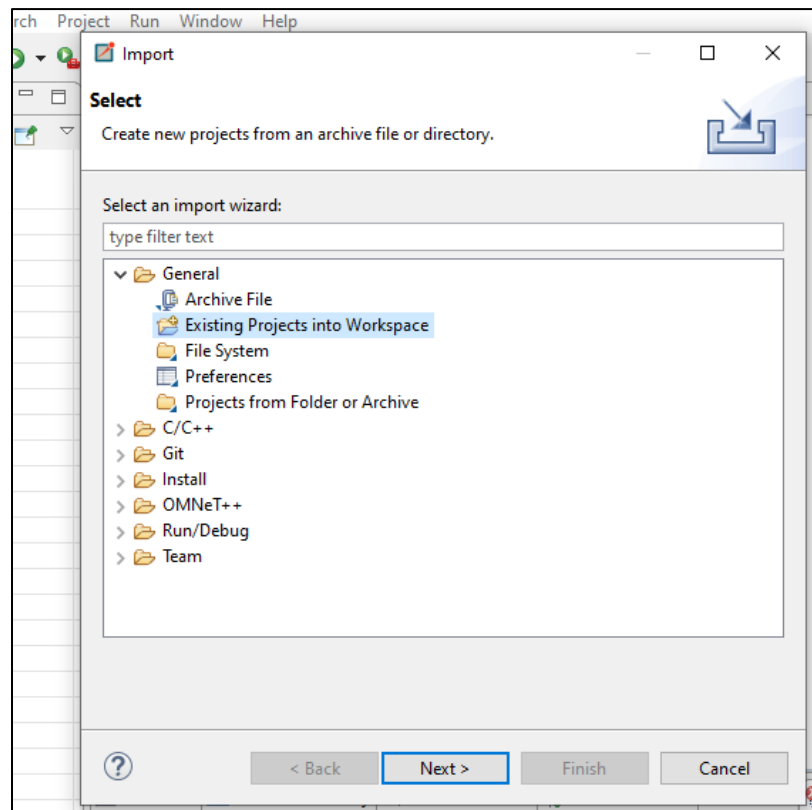
- Execute `make` and let the installation complete (this will take around 20-30 minutes).
- Once it is completed, run `omnetpp` command and a prompt will appear. Click launch in the prompt.



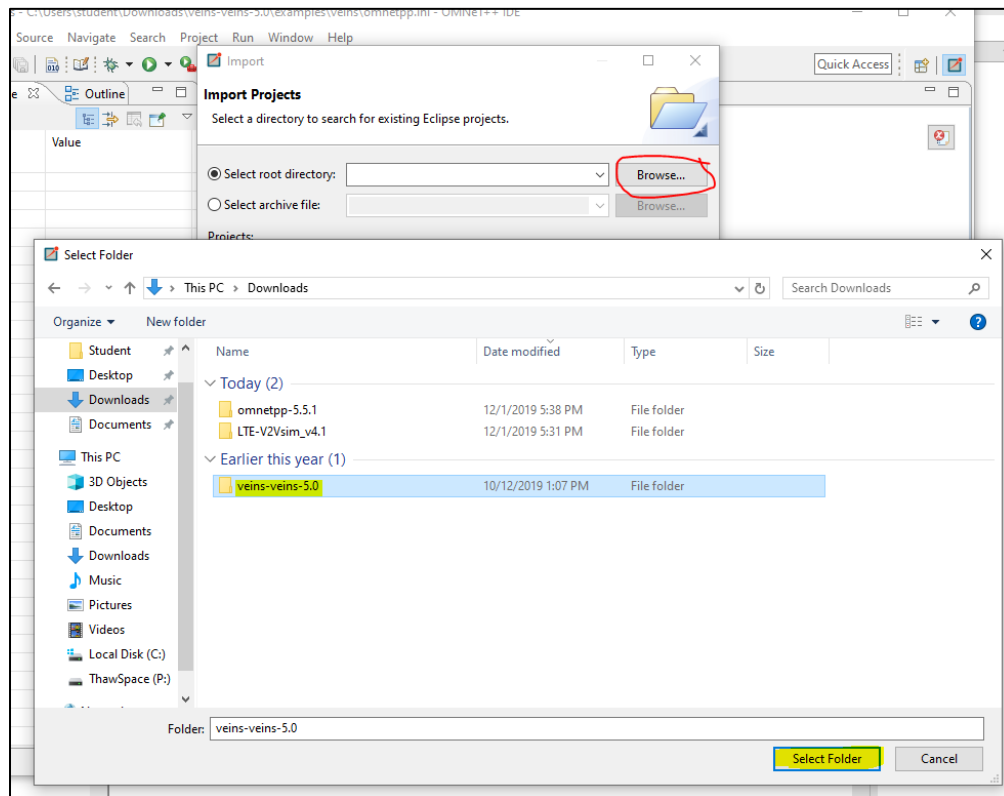
- Now, in the OMNeT++ IDE, click on the workbench icon.



- 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

```

/c/Program Files (x86)/Eclipse/Sumo/bin

/c/Program Files (x86)/Eclipse/Sumo/bin$ cd bin

/c/Program Files (x86)/Eclipse/Sumo/bin$ sumo.exe -c erlangen.sumo.cfg C:\Users\
student\Downloads\veins-veins-5.0\examples\veins\erlangen.sumo.cfg
Error: The parameter 'C:\Users\student\Downloads\veins-veins-5.0\examples\veins\erlange
n.sumo.cfg' is not allowed in this context.
Switch or parameter name expected.
Error: Could not parse commandline options.
Quitting (on error).

/c/Program Files (x86)/Eclipse/Sumo/bin$ sumo.exe -c erlangen.sumo.cfg C:/Users/
student/Downloads/veins-veins-5.0/examples/veins/erlangen.sumo.cfg
Error: The parameter 'C:/Users/student/Downloads/veins-veins-5.0/examples/veins/
erlangen.sumo.cfg' is not allowed in this context.
Switch or parameter name expected.
Error: Could not parse commandline options.
Quitting (on error).

/c/Program Files (x86)/Eclipse/Sumo/bin$ sumo.exe -c C:/Users/student/Downloads
/veins-veins-5.0/examples/veins/erlangen.sumo.cfg
Loading configuration... done.

/c/Program Files (x86)/Eclipse/Sumo/bin$ |

```

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

```

/c/Users/student/Downloads/veins-veins-5.0

Satellite.cc
Creating executable: out/clang-debug//osg-satellites_dbg.exe

Now you can type "omnetpp" to start the IDE

/c/Users/student/Downloads/omnetpp-5.5.1$ omnetpp
Starting the OMNeT++ IDE...

/c/Users/student/Downloads/omnetpp-5.5.1$ cd ../

/c/Users/student/Downloads$ dir
All_Parameters.doc      omnetpp-5.5.1          Useful_Parameters.docx
desktop.ini            omnetpp-5.5.1-src-windows.zip  veins-5.0.zip
LTE-V2Vsim_v4.1       Setup_VISSIM_x64_Uni_Full.exe  veins-veins-5.0
LTE-V2Vsim_v4.1.zip   sumo-win64-1.3.1.msi

/c/Users/student/Downloads$ cd veins-veins-5.0/

/c/Users/student/Downloads/veins-veins-5.0$ dir
configure  doxy.cfg      images      README.txt  sumo-launchd.py
COPYING   examples     Makefile    src
doc       format-code.sh  print-veins-version  subprojects

/c/Users/student/Downloads/veins-veins-5.0$ |

```

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

```

/c/Users/student/Downloads/veins-veins-5.0

/c/Users/student/Downloads/omnetpp-5.5.1$ omnetpp
Starting the OMNeT++ IDE...

/c/Users/student/Downloads/omnetpp-5.5.1$ cd ../

/c/Users/student/Downloads$ dir
All_Parameters.doc      omnetpp-5.5.1          Useful_Parameters.docx
desktop.ini            omnetpp-5.5.1-src-windows.zip  veins-5.0.zip
LTE-V2Vsim_v4.1       Setup_VISSIM_x64_Uni_Full.exe  veins-veins-5.0
LTE-V2Vsim_v4.1.zip   sumo-win64-1.3.1.msi

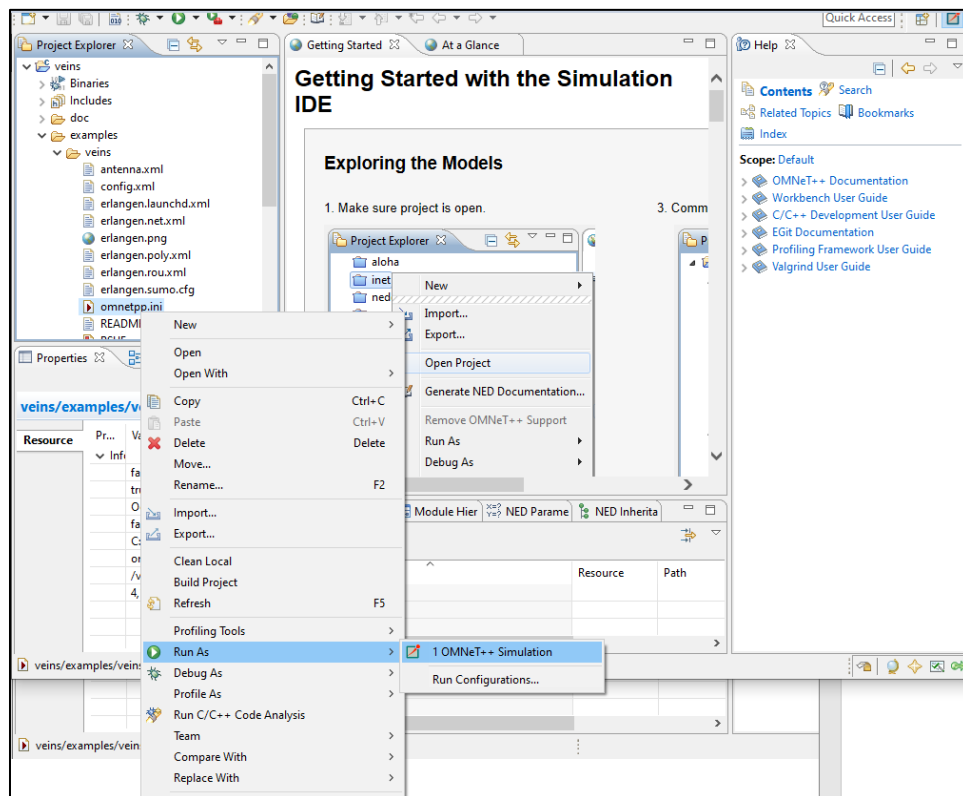
/c/Users/student/Downloads$ cd veins-veins-5.0/

/c/Users/student/Downloads/veins-veins-5.0$ dir
configure  doxy.cfg      images      README.txt  sumo-launchd.py
COPYING    examples    Makefile    src
doc        format-code.sh  print-veins-version  subprojects

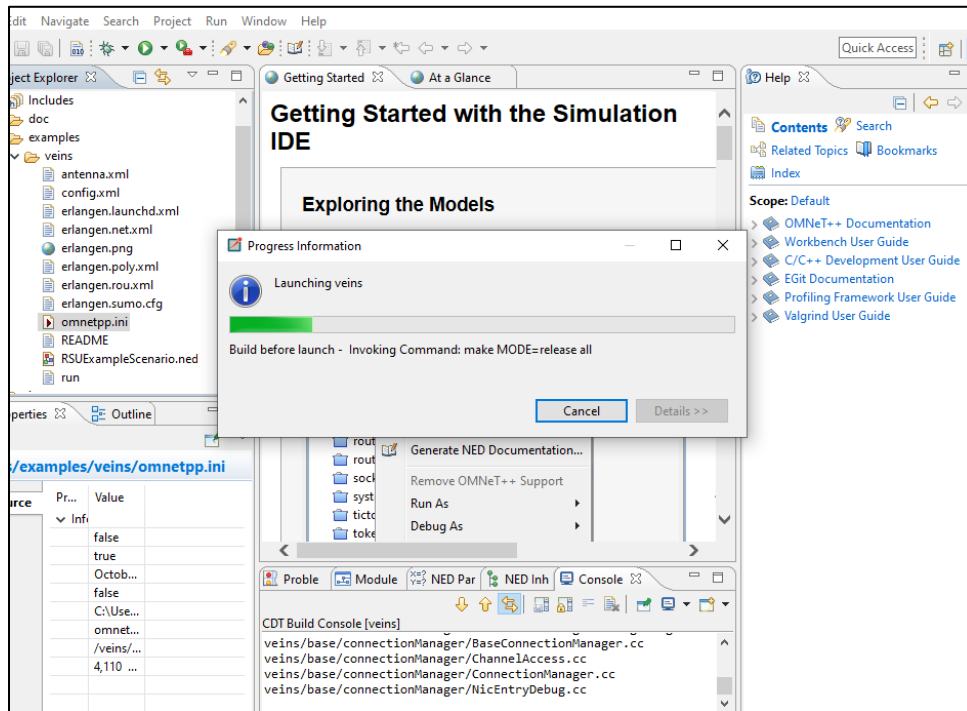
/c/Users/student/Downloads/veins-veins-5.0$ sumo-launchd.py -vv -c /c/Users/user
/src/sumo-1.2.0/bin/sumo.exe
Logging to c:/users/student/appdata/local/temp/sumo-launchd.log
Listening on port 9999

```

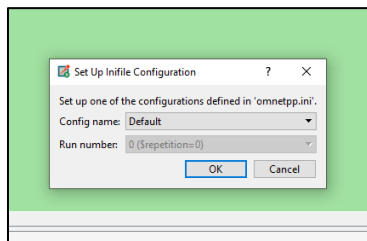
- In the OMNeT++ IDE, go to *examples>veins>omnetpp.ini*. Right click, select “Run As” and then click on “OMNeT++ Simulation.”



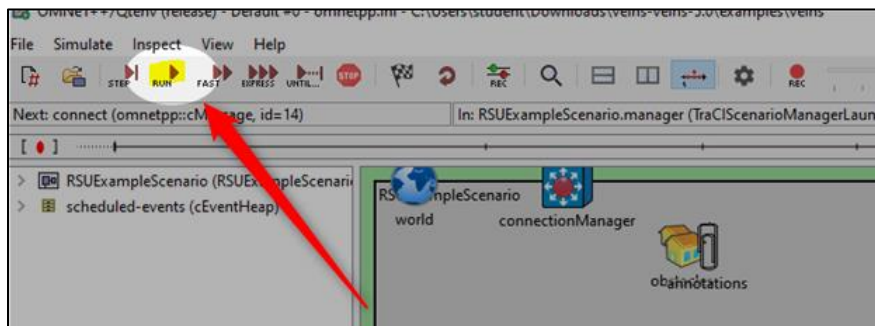
- It will load the simulation.



- After loading we get a prompt, click OK.



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

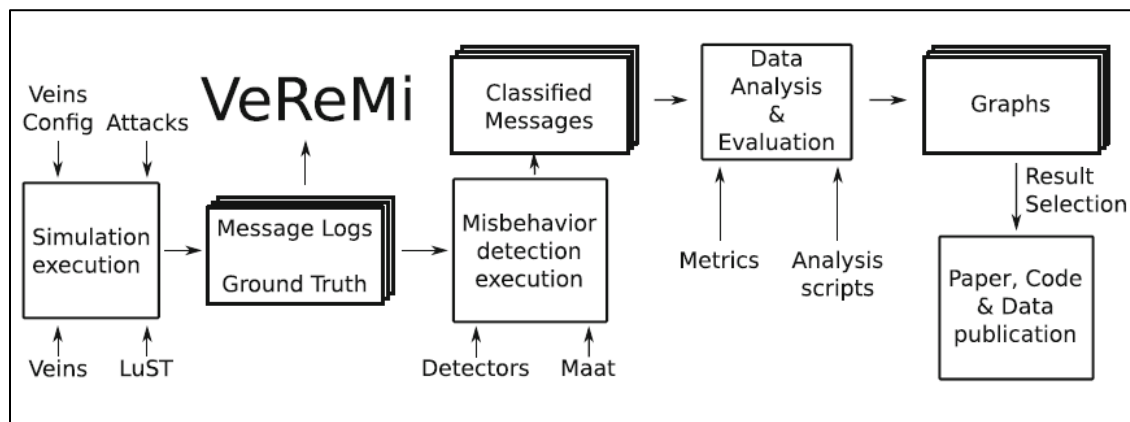


3.3. Latest versions and changelogs

- 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>

4. Vehicular Reference Misbehavior Dataset (VeReMi)

The purpose of the dataset is to provide a basis for evaluation of different misbehavior detection mechanisms, rather than a specific traffic situation that is biased towards a specific mechanism.



Maat is an evaluation framework developed by the authors for executing multiple detection algorithms simultaneously.

The dataset consists of 225 individual simulations including:

- 5 different attackers
- 3 different attacker densities
- 3 different traffic densities
- 5 repetitions for each parameter set (with different random seeds)

The attackers are:

- **The Constant Attacker:** The constant attacker transmits a fixed, pre-configured position
- **The Constant Offset Attacker:** the constant offset attacker transmits a fixed, pre-configured offset added to their actual position
- **The Random Attacker:** the random attacker sends a random position from the simulation area
- **The Random Offset Attacker:** the random offset attacker sends a random position in a preconfigured rectangle around the vehicle
- **The Eventual Stop Attacker:** the eventual stop attacker behaves normally for some time, and then attacks by transmitting the current position repeatedly (i.e., as if it had stopped)

Detection Methods:

- **The Acceptance Range Threshold (ART):** It basically uses the expected reception range as a measure for the plausibility of the position included in incoming single-hop beacon messages
- **The Sudden Appearance Warning (SAW):** It is based on the assumption that vehicles will not suddenly appear, but rather always approach from a distance; if a message originates close by with an unknown sender, it is considered malicious
- **The Simple Speed Check (SSC):** It checks if the claimed speed, position & time differences between the current and the previous beacon. If the deviation exceeds a threshold, this detector classifies the message as malicious
- **The Distance Moved Verifier (DMV):** the distance moved verifier checks whether the vehicle moved a minimum distance and if this distance is too small, the message is considered malicious.

5. 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

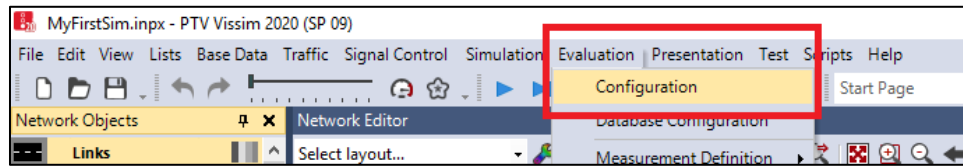
5.1. Getting started

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.

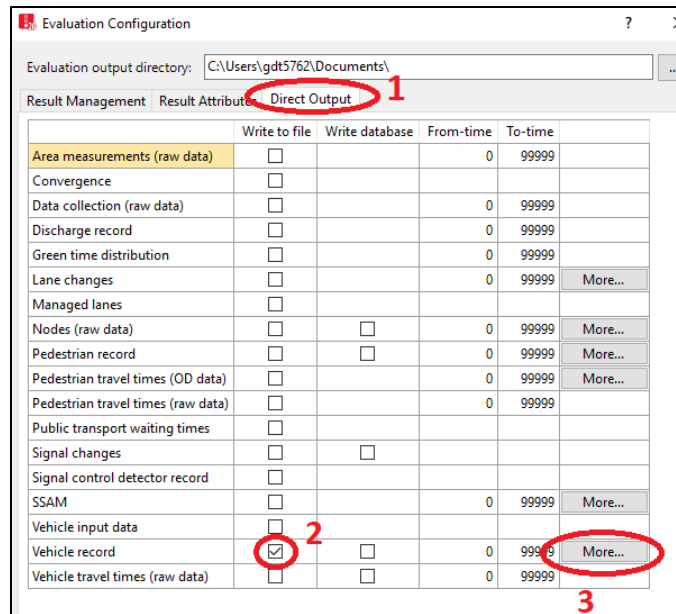
5.2. Exporting simulation data

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 be recognized by Windows, but you can read the file with a text editor like Notepad++:

```

1 FVISION
2 * File: C:\Users\gdt5762\Documents\MyFirstSim.inpx
3 * Comment:
4 * Date: 8/19/2020 5:20:48 PM
5 * PTV Vissim: 2020.00 [09]
6 *
7 * Table: Vehicles In Network
8 *
9 * SIMSEC: SimSec, Simulation second (Simulation time [s]) [s]
10 * NO: No, Number (Unique vehicle number)
11 * COORDFRONTX: CoordFrontX, Coordinate front (x) (X-coordinate of vehicle's front end)
12 * COORDFRONTY: CoordFrontY, Coordinate front (y) (Y-coordinate of vehicle's front end)
13 * COORDFRONTZ: CoordFrontZ, Coordinate front (z) (Z-coordinate of vehicle's front end) [m]
14 * SPEED: Speed, Speed (Speed at the end of the time step) [km/h]
15 * ORIENTATIONANGLE: OrientationAngle, Orientation angle
16 *
17 * SimSec:NO;CoordFrontX;CoordFrontY;CoordFrontZ;Speed;OrientationAngle
18 * Simulation second:Number;Coordinate front (x);Coordinate front (y);Coordinate front (z);Speed;Orientation angle
19 *
20 SVEHICLE:SIMSEC:NO;COORDFRONTX;COORDFRONTY;COORDFRONTZ;SPEED;ORIENTATIONANGLE
21 0.50;1;-109.551;122.897;0.000;56.36;0.25
22 0.60;1;-107.985;122.904;0.000;56.36;0.25
23 0.70;1;-106.419;122.911;0.000;56.43;0.25
24 0.80;1;-104.850;122.917;0.000;56.49;0.25
25 0.90;1;-103.280;122.924;0.000;56.56;0.25
26 1.00;1;-101.708;122.931;0.000;56.62;0.25
27 1.10;1;-100.135;122.938;0.000;56.69;0.25

```

6. Comparison of Selected V2V Simulators

	CARLA	WiLabV2XSim	VSimRTI	PTV Vissim	VEINS
License	Open Source	Open Source	License given upon request	Commercial	Temporary license given upon request
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
Platform	Python	MATLAB	Java	Windows application	OMNeT++ IDE
V2V support	No	Yes (802.11p, LTE-V2X, NR-V2X)	Yes (802.11p)	No	Yes (limited)
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	
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.
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.		

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		
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.		
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.		
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.