The open source vehicular network simulation framework.

Tutorial

Learn how to set up Veins.

Note: Veins runs on Linux, Mac OS X, and Windows. Because of the extensive debugging capabilities that it offers, Veins is best built and run on Linux.

For building on Linux, some packages may need to be installed. On Ubuntu Linux, this will likely mean running

```
sudo apt-get install build-essential gcc g++ bison flex perl tcl-dev tk-dev blt libxml2-dev zlib1g-dev default-jre doxygen graphviz libwebkitgtk-1.0-0 openmpi-bin libopenmpi-dev libpcap-dev autoconf automake libtool libproj0 libgdal1-dev libfox-1.6-dev libgdal-dev libxerces-c-dev
```

to install them.

On Mac OS X, this will likely mean installing equivalent packages via Macports by running

```
sudo port install bison zlib tk blt libxml2 libtool xercesc proj gdal fox
```

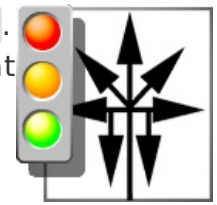
to install them.

The OMNeT++ install guide has many helpful hints on pre- and post-configuration of your system.

This tutorial assumes that you are using Windows 7, that your home directory is `C:\Users\user`, and that all necessary software will be installed in `C:\Users\user\src` (which was already created). Aside from the paths given and the opening of the OMNeT++ MinGW command line window vs. a regular command line window, these steps are similar when building and running the simulations on Linux

or Mac OS X.

Step 1: Download SUMO



Download the SUMO 0.21.0 binaries and unpack them as `C:\Users\user\src\sumo-0.21.0`. This should give you an executable `C:\Users\user\src\sumo-0.21.0\bin\sumo.exe`. Note that recent versions of the SUMO binaries require the Microsoft Visual C++ 2010 Redistributable Package (x86) to be installed.

If you want to (or have to) build SUMO manually, detailed instructions for many platforms can be found on the Installation Instructions on the SUMO website.

Note: Make sure you are running the right version of SUMO for your Veins installation. If you are unsure which version of SUMO you are running, the output of `sumo --version` will tell you. You can get a quick overview of supported SUMO versions from the Veins Changelog.

Step 2: Download and build OMNeT++ 4

Download OMNeT++ 4.4 for Windows and unpack it as `C:\Users\user\src\omnetpp-4.4`.

Note: If you unpack OMNeT++ to a different folder, make sure it contains no spaces.

This should give you a script

```
C:\Users\user\src\omnetpp-4.4\mingwenv.cmd
```

that you can run to open a MinGW command line window, which closely mimics a Linux environment.

Build OMNeT++ 4 by running

```
./configure
```

(making sure to examine the summary for potential errors) followed by

```
make
```

to start the build process. If all went well, this will result in `/c/Users/user/src/omnetpp-4.4/bin/omnetpp` being built. Run

```
omnetpp
```

to launch the OMNeT++ 4 IDE. This tutorial will assume that you picked `C:\Users\user\src\omnetpp-4.4\samples` as your workspace.

Step 3: Download and build the Veins module framework

Download Veins 3.0 and unpack it as `C:\Users\user\src\veins-3.0`. Import the project into your OMNeT++ IDE workspace by clicking `File > Import > General: Existing Projects into Workspace` and selecting the directory you unpacked the module framework to.

Build the newly imported project by choosing `Project > Build All` in the OMNeT++ 4 IDE. After the project built, you are ready to run your first IVC evaluations, but to ease debugging, the next step will ensure that SUMO works as it should.

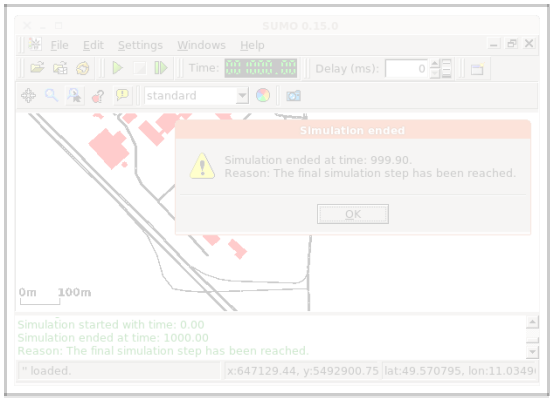
Step 4: Make sure SUMO is working

In the OMNeT++ MinGW command line window, you should be able to have SUMO simulate an example scenario by changing the current directory to `/c/Users/user/src/veins-3.0/examples/veins/` using `cd ../veins-3.0/examples/veins` and running

```
/c/Users/user/src/sumo-0.21.0/bin/sumo.exe -c erlangen.sumo.cfg
```

to start SUMO. You should see a line saying "Loading configuration... done.", then a simulation time step counter running from 0 to 1000 and disappearing again.

To get an impression of what the example scenario looks like, you can also run it using `sumo-gui.exe`, but this is not required for Veins to work.



Example scenario running in the SUMO GUI.

Final step: Run the Veins demo scenario

To save you the trouble of manually running SUMO prior to every OMNeT++ simulation, the Veins module framework comes with a small python script to do that for you. In the OMNeT++ MinGW command line window, run

```
/c/Users/user/src/veins-3.0/sumo-launchd.py -vv -c /c/Users/user/src/sumo-0.21.0/bin/sumo.exe
```

to start it. This script will proxy TCP connections between OMNeT++ and SUMO, starting a new

copy of the SUMO simulation for every OMNeT++ simulation connecting. The script will will print

```
Listening on port 9999
```

and wait for the simulation to start.

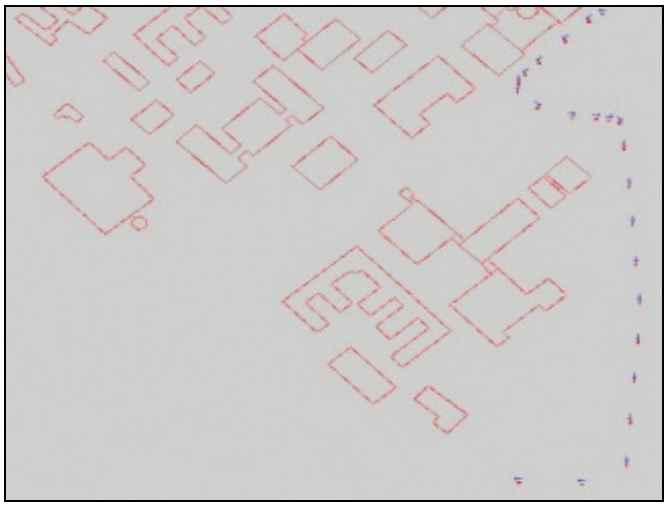
Hint: If you don't want to enter the full path to your sumo binary everytime you start the launchd, you can also add it to the PATH variable of your mingw environment. To do so, open msys/etc/profile in your OMNeT directory and add

```
export PATH=$PATH:/c/Users/user/src/sumo-0.21.0/bin/
```

In the OMNeT++ 4 IDE, simulate the Veins demo scenario by right-clicking on `veins-3.0/examples/veins/omnetpp.ini` and choosing `Run As > OMNeT++ simulation`. Don't forget to allow access to SUMO through any personal firewall you might run. Similar to the last example, this should create and start a launch configuration. You can later re-launch this configuration by clicking the green `Run` button in the OMNeT++ 4 IDE.

If everything worked as intended this will give you a working simulation scenario using OMNeT++ and SUMO running in parallel to simulate a stream of vehicles that gets interrupted by an accident.

If you select the configuration named `debug` when starting the simulation, you will see a wealth of debug output and visual annotations such as building positions on the canvas, albeit at much reduced speed. In this configuration, it is best to turn off module names (via the toolbar button or by pressing Ctrl+D).



GUI screencast of information dissemination

via flooding while buildings block transmissions.

This concludes the mini-tutorial. In-depth information on how to use Veins is available in the documentation, with answers to the most common questions in the list of Frequently Asked Questions (FAQ).
