SDAV: Simulation

1. Artery Simulator

Artery is a V2X simulation framework for ETSI ITS-G5 protocols like GeoNetworking and BTP. Like many VEINS-based simulators, Artery is a co-simulation of networking (handled by Artery) and a physical representation of the vehicle (handled by SUMO).

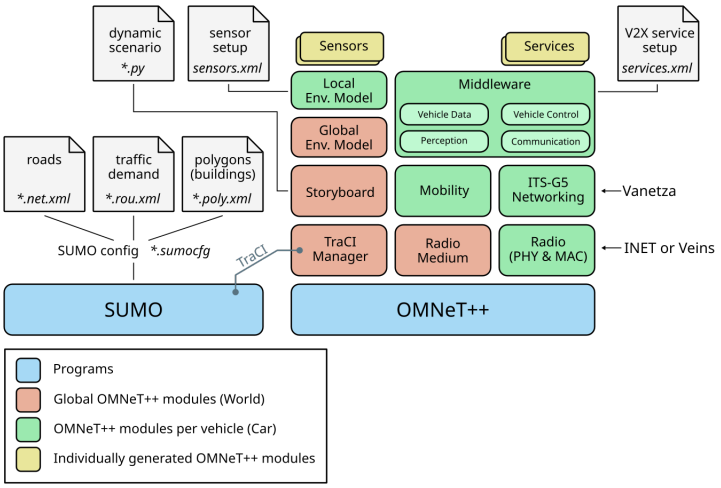

Figure 1: Artery main components

Figure 1 shows the Artery architecture main components. The SUMO simulation is configured by a *.sumocfg file that calls on the roads (*.net.xml), traffic demand (*.rou.xml) and buildings (*.pol.xml) configuration files. So SUMO runs an independent simulation in parallel to Artery. Artery is composed by a variety of modules. Some handling the ETSI compliant networking layers and others handling data acquisition from sensors and/or vehicular mobility.

The central component of vehicles is the Middleware module. The middleware creates service modules according to an XML configuration file provided by the user. It is possible to equip vehicles with different sets of applications by configuration, i.e. communication capabilities can vary among vehicles.

1.1. Installation

The simulation is available in the following repository Gitlab under the artery_sim branch.

You must first install SUMO, Artery and OMNeT++ independently before downloading the branch. Please use the links bellow:

- Install Omnetpp (use version 5.6) https://omnetpp.org/download/old ; Guide: https://doc.omnetpp.org/omnetpp/InstallGuide.pdf
- Install SUMO from source (version 1.11.0) https://sumo.dlr.de/releases/1.11.0/ ; Guide: https://sumo.dlr.de/docs/Installing/Linux_Build.html
- Install Artery: http://artery.v2x-research.eu/install/

Once the libraries are installed, do remember to test the Artery installation by running the command:

```
cmake --build build --target run_example
```

Once Artery is working you can set up the Gitlab repository inside the Artery scenarios using the following instructions:

- clone this repository in another folder and "git checkout artery_sim" this branch
- rename ".git" folder at artery code root dir to ".git_artery"
- copy&paste ".git" from sdav to artery code root dir
- now git will use .git from sdav (which ignores artery folders)
- do a git status to test if it works
- do a "git pull"

Dependencies:

- nlohmann json parser (https://github.com/nlohmann/json/tree/develop) — Download branch and extract nlohmann from "single_include/json.hpp" file to the monitorAV folder

1.2. Tutorial References

Generating maps and defining traffic through SUMO has many tutorials available through the following website: https://sumo.dlr.de/docs/Tutorials/

Artery has little in terms of tutorials, but the API has many self-explanatory components and the scenario directory has many usage examples found here: https://github.com/riebl/artery/tree/master/scenarios

1.3. Step-by-Step Tutorial

In this step by step tutorial we are going to generate a circular road with a middle point. This includes the generation of all SUMO configuration files and how to select this simulation in Artery.

With SUMO installed, the net edition tools should be available in the command line. If they is not available you can find them in the $SUMO_HOME/bin directory.

1. Execute the following command in the command line
   ```
   netgenerate --spider --spider.arm-number=100 --spider.circle-number=1 --spider.space-radius=110 --spider.omit-center --no-turnarounds --output-file=<filename>.net.xml
   ```
2. This generates a circular network with a radius of 110 meters divided into 100 segments.
3. Open the netedit tool through the command line and open the created *.net.xml file. You should see Figure 2:

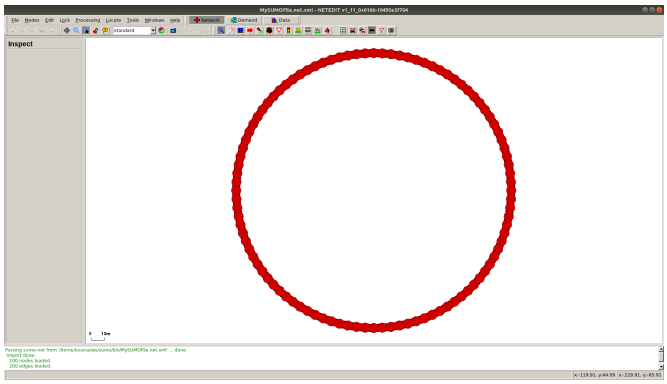

Figure 2: netgenerate file open in netedit.

1. SUMO represents roads and traffic through graph nodes. When we need to create a roundabout, or a circular road with a specific geometry, the netedit tools can be used to modify and adapt roads.
2. SUMO does not have true curves, it uses linear approximation of curves. If you zoom into the road we created you can see a collection of a 100 straight edges approximating a circle.

4. Clicking on the demand button you can get a clearer view of the road (Figure 3).

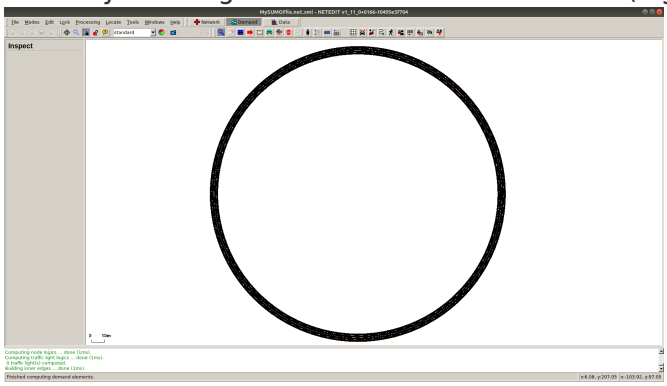

Figure 3: Demand tab view of circular road

5. Select network mode again, then select edge mode, then select edit ->Create consecutive edges.

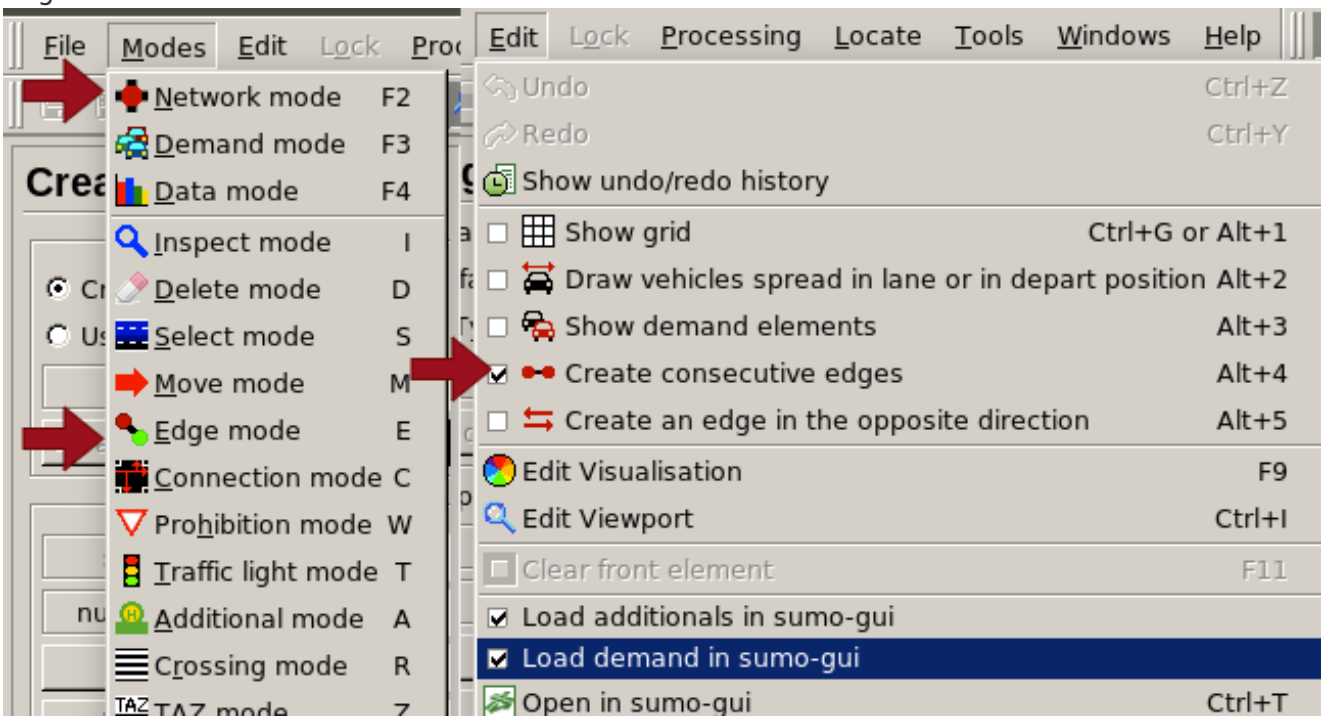

Figure 4: Selecting network mode and creating consecutive edges

6. You can then connect any edge you want (or create new ones) like in Figure 5:

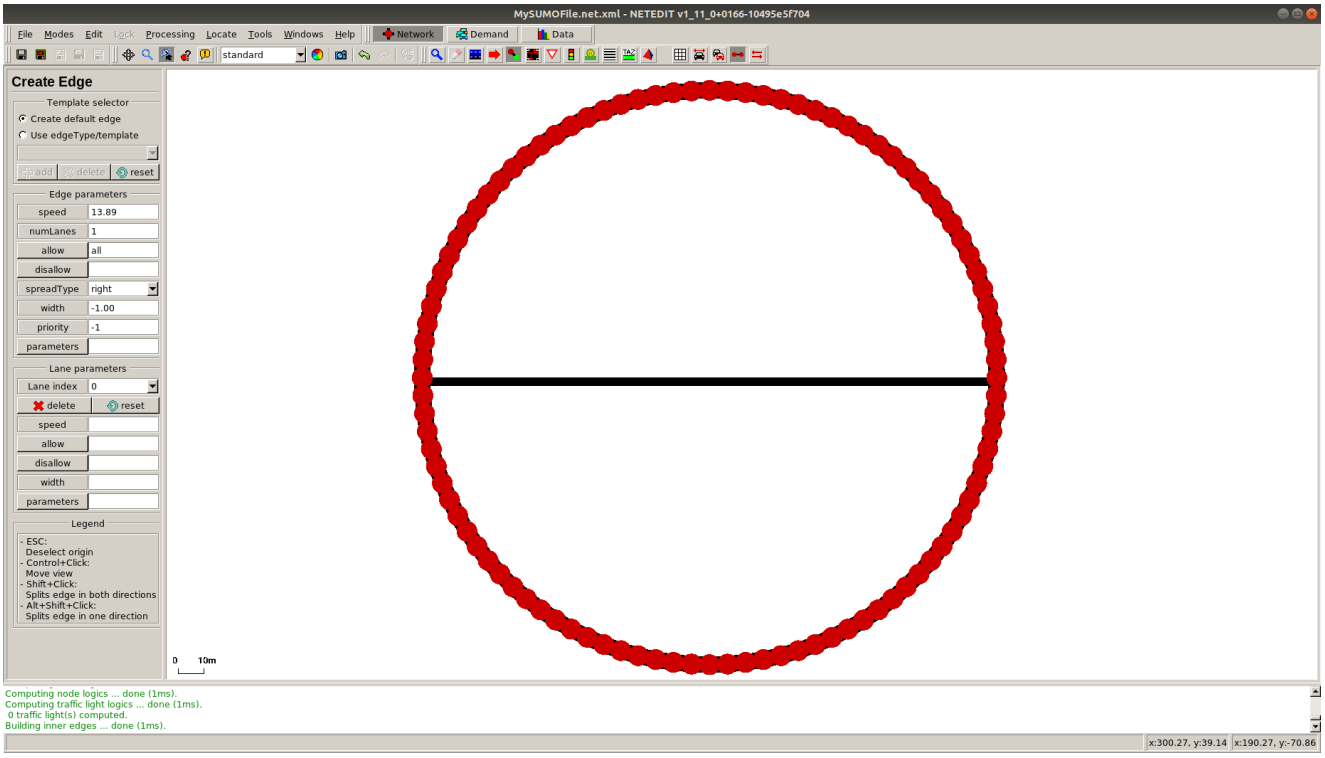

Figure 5: Edge connection

7. Now return to demand mode and select route mode.
8. Select any edge of the map to create start point of the route and select another edge to add a waypoint to the route. If you wish you can select as many waypoints as you want to generate a longer route.
    1. In SUMO all vehicles have some form of predefined route. Routes are stored in the *.rou.xml file and can be associated with any number of vehicles. Vehicles can also have personal routes created during their definition.
    2. When Artery is receiving Motion Vectors from the IOT it tries to associate the geolocation with the map and creates a route for the vehicle to follow.
9. With the route selected you can click the "Finish route creation" button to the side
10. Select vehicle mode and click anywhere on the route once to add a vehicle.
11. You can now "save this network as" and save it in a folder inside the Artery scenario to save your *.net.xml file.
12. Also save the Demand files through File->Demand Elements->Save Demand Elements As to save your *.rou.xml file
13. Select Edit->Open in sumo-gui
14. Once open you can simply select File->Save Configuration to save your *.sumocfg file.
    1. To run this simple simulation and be able to see the vehicle, you can set the "Delay (ms)" to anything above 80 ms. This adds a small delay between simulation steps and allows us to see the vehicle moving, otherwise the simulation executes as fast as possible and ends.
    2. On an Artery simulation you can leave the "Delay (ms)" field at 0 (zero).
15. You can now close both sumo-gui and netedit
16. Now select your *.net.xml file and look for the projParameter. Substitute "!" for "+proj=tmerc +ellps=WGS84 +datum=WGS84 +units=m +no_defs".
    1. The projection parameter tells SUMO how to calculate the latitude-longitude projection of the cartesian xy position of objects inside the simulation. The configuration above simply defines the projection as that of the WGS84 format. You can find an in-depth explanation here.
17. Add the following to the configuration section of your *.sumocfg file:

<time> <begin value="0"/> <step-length value="0.1"/> </time>

18. At this point you should have a *.sumocfg, a *.net.xml and a *.rou.xml file in a folder inside your Artery scenarios.

With the SUMO simulation configured, we can use the Artery simulation to collect the Motion Vectors of the vehicle.

1. Change the following lines inside the omnetpp.ini file to set the correct *.sumocfg and to turn off the creation of RSUs:

```
16 *.traci.core.startTime = 0s ... 20 *.traci.launcher.sumocfg = "../<folder name>/<file name>.sumocfg" 21 *.traci.launcher.sumo = "sumo-gui" ... 27 #*.staticNodes.nodes = xmldoc("grid_5x5/RSUs.xml")
```

2. Inside the services.xml file keep only the code below, the rest can be commented or temporarily removed.

```
<?xml version="1.0" encoding="utf-8"?> <services> <service type="MonitorService" name="MonServ"> <listener port="2002" channel="180" aid="36"/> <filters><name pattern="ego0"/></filters> </service> </services>
```

3. Execute the simulation through:

```
cmake --build build --target run_monitorAV
```

4. Once the build is complete, the simulation window should pop up. Click run to start, then click run again once the sumo-gui window pops up.
5. You can then see the Motion Vector of the vehicle in the log at scenarios/monitorAV/results/monitor_car-ego0.out.

The output format of the log is one motion vector per line in a json format for ease of reading:

```
"Motion_Vector":{"pos":[140.000000,8770.000000,800001.000000],"speed":0.000000,"acceleration":0.000000,"yaw_rate":0.000000,"heading":2700.000000,"dim":[1.800000,5.000000,0.000000],"class":5,"id":0,"conf_percent":100,"timestamp":1720332}
```

The vehicle is stopped at a longitude and latitude, the altitude equals the AltitudeValue_unavailable in Artery and can be ignored. The vehicle is facing east with a 270 degrees heading and has the regular dimensions of 1.8 meters width and 5 meters length.

As the vehicle begins moving we can visualize the increase in acceleration and speed which, as the vehicle reaches target speed turns into a controlled acceleration/deceleration.

And once the vehicle inititates the first curve we can see the heading start to change:

```
"pos":[140.000000,-9010.000000,800001.000000],"speed":7.281056,"acceleration":2.600000,"yaw_rate":0.000000,"heading":2700.000000
```

"pos":[140.000000,-9080.000000,800001.000000],"speed":7.541056,"acceleration":2.600000,"yaw_rate":0.000000,"heading":2700.000000
"pos":[140.000000,-9150.000000,800001.000000],"speed":7.630857,"acceleration":0.898008,"yaw_rate":0.000000,"heading":2700.000000
"pos":[140.000000,-9210.000000,800001.000000],"speed":7.180857,"acceleration":-4.500000,"yaw_rate":0.000000,"heading":2700.000000
"pos":[150.000000,-9270.000000,800001.000000],"speed":6.730857,"acceleration":-4.500000,"yaw_rate":-800.000000,"heading":2708.000000
"pos":[160.000000,-9330.000000,800001.000000],"speed":6.658843,"acceleration":-0.720143,"yaw_rate":-1000.000000,"heading":2718.000000
"pos":[170.000000,-9390.000000,800001.000000],"speed":6.664986,"acceleration":0.061432,"yaw_rate":-1000.000000,"heading":2728.000000
"pos":[180.000000,-9450.000000,800001.000000],"speed":6.713585,"acceleration":0.485997,"yaw_rate":-1200.000000,"heading":2740.000000
"pos":[210.000000,-9500.000000,800001.000000],"speed":6.649595,"acceleration":-0.639905,"yaw_rate":-3900.000000,"heading":2779.000000
"pos":[240.000000,-9550.000000,800001.000000],"speed":6.652124,"acceleration":0.025289,"yaw_rate":-4000.000000,"heading":2819.000000
"pos":[270.000000,-9600.000000,800001.000000],"speed":6.660485,"acceleration":0.083612,"yaw_rate":-4200.000000,"heading":2862.000000

Unfortunately in SUMO the yaw rate is not the driver of the change in heading, but a measurement of the result. As the vehicle changes from one edge to the other, the VehicleDataProvider class in Artery calculates the yaw rate that would result in the heading change.

2. Motion Vector Datasets

There are some Motion Vector datasets available that can be used in combination with the simulation:

1. VeReMi: A large MV dataset with introduced faults for misbehavior detection. Motion vectors are incomplete.
2. F2MD: A misbehavior detection simulation based on VEINS. Can be used to generate the VeReMi dataset and control the types of faults being injected. Produces a complete Motion Vector.