

VANET Simulation Using FOSS

Networking is a growing area of research wherein a lot has been explored, but a lot of avenues are still wide open. The latest research trend in this domain is VANET (Vehicular Ad-hoc Network).

Part—1

Networking is a growing area of research wherein a lot has been explored, but a lot of avenues are still wide open. The latest research trend in this domain is VANET (Vehicular Ad-hoc Network).

As the name suggests, VANET is a network wherein the nodes are vehicles equipped with on-board units (OBU) for networking. A VANET may also have Road-Side Units (RSU), which are static infrastructure nodes to support the network's functionality. VANET communication comes in two flavours: Vehicle-to-Vehicle (V2V) communication and Vehicle-to-Infrastructure (V2I) communication. A typical VANET scenario is shown in Figure 1 and the VANET architecture can be seen in Figure 2.

Although it is a type of mobile ad-hoc network, VANET exhibits its own distinctive characteristics, such as:

- The nodes move at very high speed.
- It has a highly dynamic topology with very high node density.
- The network is frequently disconnected.

- Mobility patterns are constrained.
- There are hard delay constraints.
- It involves interaction with on-board sensors.

Some of the applications of VANET to avoid collisions include traffic signal violation warnings, stop sign violation warnings, left turn assistance, stop sign movement assistance, intersection collision warnings, blind merge warnings, pedestrian crossing information, etc. There are many such applications, ranging from safety-critical features to infotainment. The research done in this area cannot be tested 'live' on the road, due to the safety implications. We need to set up a simulation environment for such network testing. There are two components of VANET simulation:

- The mobility component: Defining road topology and vehicle movement
- The networking component: Network traffic simulation

There are various VANET simulators available, as shown in Figure 3. This article focuses on the open source Network Simulator-2 (NS2) to simulate the networking component,

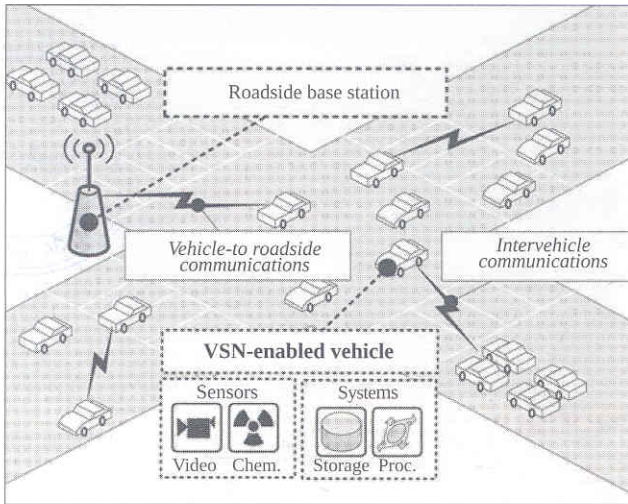


Figure 1: A typical VANET scenario

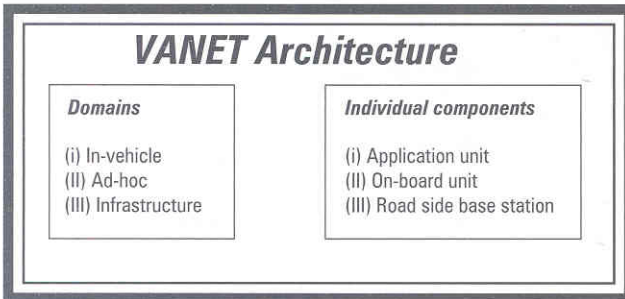


Figure 2: VANET architecture

and Simulation of Urban Mobility (SUMO) to simulate the mobility component.

For the identification of a traffic simulator, the following parameters are considered:

- Accurate and realistic topological maps: geographical maps obtained from TIGER, GDF, GPSTrack, Google Maps or ESRI.
- The database.
- Obstacles: The hurdles to a vehicle’s mobility and communication.
- The smoothness of deceleration and acceleration.
- Human driving patterns.
- Intersection management, which is based on some stochastic model.
- Time patterns: The consideration of peak hours and time constraints.
- External influences, for example, road maintenance, traffic conditions and accidents.

Considering the aspects mentioned here, SUMO can be said to have most of the desired features for mobility simulation with visualisation support. SUMO helps in generating the first phase of VANET simulation, wherein we generate the road map and the vehicles’ flow, which is known as a mobility pattern. The files needed to generate a mobility pattern using SUMO are:

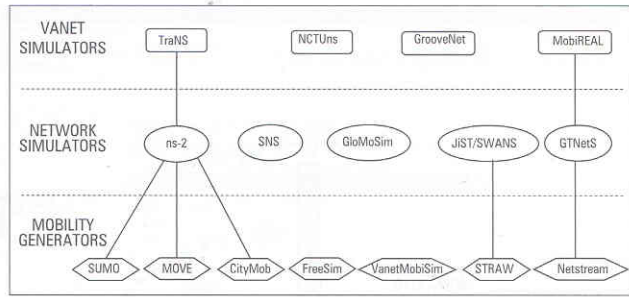


Figure 3: VANET simulators

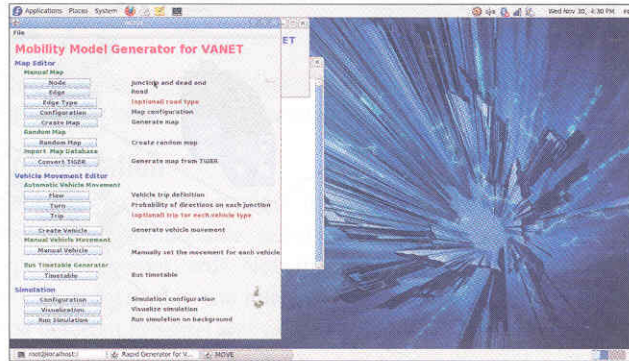


Figure 4: MOVE main screen for mobility pattern generation

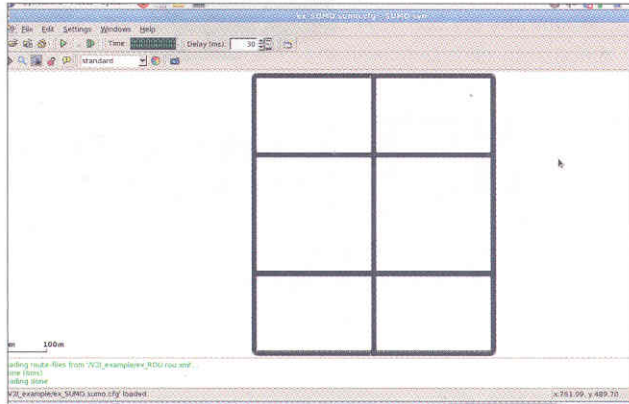


Figure 5: Sample SUMO output

1. <Filename>.nod.xml: Defines nodes
2. <Filename>.edge.xml: Defines edges
3. <Filename>.netc.cfg
4. <Filename>.net.xml: Generated using the net-convert command, from the above two files
5. <Filename>.ow.xml: Defines traffic flow definition
6. <Filename>.rou.xml: Defines vehicle routes
7. <Filename>.sumo.cfg: Simulation configuration

Generating so many files for a large number of vehicle nodes is cumbersome, so instead of typing the file contents in the specific formats, we can use another open source project called MObility generation for Vehicular Environment (MOVE). This provides a GUI to generate SUMO outputs—also, the defined road mobility is converted to NS2-readable form (.tcl). NS2 then can be used to simulate the networking-

(Continued on Page 89....)

(Continued from Page 84....)

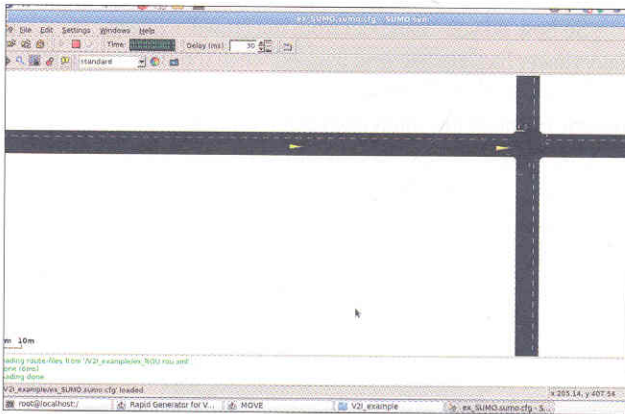


Figure 6: A close look at SUMO output

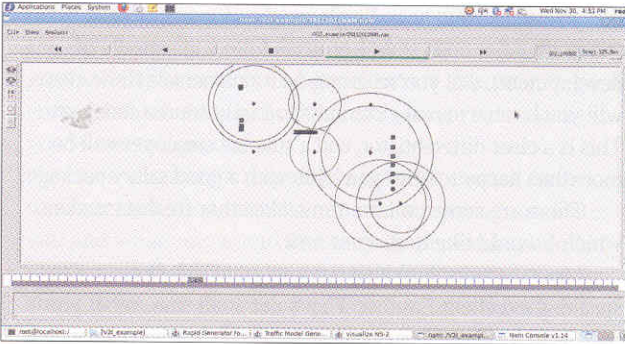


Figure 7: Sample of NS-2 animation


related aspects. Figure 4 shows the main screen of MOVE for vehicular traffic and road-map generation. Figures 5 and 6 display a sample SUMO output.

Once the mobility pattern is generated, MOVE converts it into the trace files readable by NS-2 for network traffic simulation. Figure 7 shows the NS-2 animation of the mobility pattern in Figure 5.

A synopsis of the simulation process

The process involves:

- Defining the nodes and edges in the desired road topology, in the `.nod.xml` and `.edge.xml` files respectively. This can be done using the MOVE GUI, or by simply creating and editing a file.
- Next, we generate the road map, i.e., `.net.xml`, using the `net-convert` command or MOVE.
- Define the vehicle flow in the `.flow.xml` file, and generate `.rou.xml` and `sumo.cfg` files, using MOVE.
- Visualisation of the road traffic configuration can be done using `sumo-gui`.
- Now `.tr`, `.tcl` and `.nam` file can be generated, using MOVE, from the road traffic simulation.
- Necessary changes are made in the `.tcl` file for relevant network simulation or analysis.

In the next article, I plan to cover step-by-step simulation generation using SUMO and MOVE. And subsequently, I aim to share NS-2 simulation for wired and wireless networks on Fedora 14, or Ubuntu, from scratch. **END** 

By: Pooja P Shah

The author is an assistant professor at Nirma University, Gujarat and a researcher in the field of Vehicular Ad-hoc Networks. Her hobbies are teaching, learning new technologies, listening to music and writing.