Seminar at UniRC – Jul 2012

Tutorial:
Simulating VANET and ITS
(using OMNeT++ and SUMO)

MJ (Thinus) Booysen
mjbooysen at sun dot ac dot za
Presentation Overview

- History and background
- Why is VANET special?
- Vehicular networking vs. ITS
- What is required for simulation?
- Existing approaches
- Simulation packages
- Veins (OMNeT++ & SUMO)
- SUMO demo
- OMNeT++ demo
- Veins demo
- Remaining challenges
History and Background

- Traditional traffic simulations:
  - Civil worked with vehicular traffic
    - Road design
    - Intersection
  - E&E worked with network traffic
    - Computer networks
    - Mobile networks

- Recent developments:
  - Vehicular Ad-hoc NETworks (VANET)
  - Intelligent Transportation Systems (ITS)
Safety and traffic management  
Entertainment and Internet connectivity

- OBU (On Board Unit)
- RSU (Road Side Unit)
- V2I (Vehicle to Infrastructure)
- V2V (Vehicle to vehicle)
- Wi-Fi
- CAN (Controller Area Network)
- ISP (Internet Service Provision)
- ITS (Intelligent Transportation Systems)
- MM (MOST Master)
- WIMAX (Worldwide Interoperability for Microwave Access)
- WAVE (Wireless Access in Vehicular Environments)

- Operators and end users
- Servers and databases
- ITS network
- Internet
- RSU (Wi-Fi Cellular)
- WIMAX
- WAVE
- Servers and Consumers
- Networks
- Intra-car
- Extra-car
- V2I
- V2V

- CAN bus
- Navigation
- Emergency detectors
- Audio-visual sources
- Displays
- MOST ring

- Operators and end users
- Servers and databases
- ITS network
- Internet
- RSU (Wi-Fi Cellular)
- WIMAX
- WAVE
- Servers and Consumers
- Networks
- Intra-car
- Extra-car
- V2I
- V2V

- CAN bus
- Navigation
- Emergency detectors
- Audio-visual sources
- Displays
- MOST ring

- Operators and end users
- Servers and databases
- ITS network
- Internet
- RSU (Wi-Fi Cellular)
- WIMAX
- WAVE
- Servers and Consumers
- Networks
- Intra-car
- Extra-car
- V2I
- V2V

- CAN bus
- Navigation
- Emergency detectors
- Audio-visual sources
- Displays
- MOST ring

- Operators and end users
- Servers and databases
- ITS network
- Internet
- RSU (Wi-Fi Cellular)
- WIMAX
- WAVE
- Servers and Consumers
- Networks
- Intra-car
- Extra-car
- V2I
- V2V

- CAN bus
- Navigation
- Emergency detectors
- Audio-visual sources
- Displays
- MOST ring

- Operators and end users
- Servers and databases
- ITS network
- Internet
- RSU (Wi-Fi Cellular)
- WIMAX
- WAVE
- Servers and Consumers
- Networks
- Intra-car
- Extra-car
- V2I
- V2V

- CAN bus
- Navigation
- Emergency detectors
- Audio-visual sources
- Displays
- MOST ring
Vehicular simulation: challenges, what’s different?

- V2I and V2V
- Mobility patterns
  - Fast
  - Unpredictable, but constrained.
- Broad range of applications
  - Safety critical
  - Infotainment
- Power usage
What is required for simulation?

- **VANET:**
  - Accurate simulation of communications
    - signal propagation
    - networking
  - Accurate simulation of vehicle mobility
  - Comms simulation use mobility info

- **ITS:**
  - Same as VANET
  - More applications support
  - Comms simulation (application) affects mobility
Some approaches

- **Mobility**
  - Fixed number of vehicles at constant speed in straight line in the same direction
  - Proprietary simulators (simple following, no overtaking)
  - Generate traces with a proper simulator, save to file
  - Limited scenarios and densities
    - Highway
    - Urban

- **Communications**
  - Fixed range, if within range communications successful.
  - Taking into account radio properties
  - Interference
  - Obstacles
## VANET simulation packages

<table>
<thead>
<tr>
<th>Tool</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trafic and Network Simulator <em>(TraNs)</em> (SUMO and ns-2)</td>
<td>Flexibility and real world maps. Integrated solution.</td>
<td>No feedback is provided from ns-2 to SUMO. Development suspended recently.</td>
</tr>
<tr>
<td>National Chiao Tung University network simulator <em>(NCTUns)</em></td>
<td>Single application with integrated GUI. Popular for VANET research.</td>
<td>NCTUns is UNIX-based and only runs on Fedora. Limited support.</td>
</tr>
<tr>
<td><strong>VanetMobiSim</strong> with <strong>ns-2</strong>.</td>
<td>Flexible mobility models with micro-mobility and macro-mobility models. Maps can be imported from TIGER database.</td>
<td>No feedback is provided from ns-2 to VanetMobiSim. Separate simulators.</td>
</tr>
<tr>
<td>Vehicles in Network Simulation <em>(Veins)</em>, which integrates SUMO and OMNeT++</td>
<td>Flexibility and real world maps. Active community with support. Full IEEE 802.11p implementation. Integrated solution.</td>
<td></td>
</tr>
</tbody>
</table>
Veins (SUMO & OMNeT) Simulation model

Veins: veins.car2x.org/ (easy installation here)
OMNeT++: www.omnetpp.org/ (easy thorough tutorial here)
MiXiM: mixim.sourceforge.net/
SUMO: sumo.sourceforge.net/
OpenStreetMap: www.openstreetmap.org/
SUMO Setup: Map: www.OpenStreetMap.org

Export map from web site, and convert to XML with SUMO’s netconvert:

```
netconvert --osm rc.osm
```
SUMO setup: trips

Generate random trips (from A to B) from the road network (net.net.xml)

randomTrips.py -n net.net.xml -l -e 600 -o trips.trips.xml

Convert the trips to routes (what is between A and B) and traffic flow

duarouter -n net.net.xml -t trips.trips.xml -o routes.rou.xml --ignore-errors
SUMO setup: simulation

Can also specify flows (in stead of single vehicles), vehicle types, different following models, etc.

```xml
<vType id="vtype0" accel="2.6" decel="4.5" sigma="0.5" length="2.5" minGap="2.5" maxSpeed="35" color="1,1,0"/>
<flow id="flow0" type="vtype" route="route0" begin="0" period="5" number="200" departlane="random" departpos="base"/>
```

Configure the simulation by specifying the network, the routes, and the duration in a config file.

```xml
<configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:noNamespaceSchemaLocation="http://sumo.sf.net/xsd/sumoConfiguration.xsd">
  <input>
    <net-file value="net.net.xml"/>
    <route-files value="routes.rou.xml"/>
  </input>

  <time>
    <begin value="0"/>
    <end value="400"/>
  </time>
</configuration>
```

sumo.sumo.cfg
SUMO simulation
OMNeT++ Setup

- OMNeT provides the messaging platform for simulation.
- OMNeT is a modular discrete event simulation environment that can be used to simulate nearly anything.
- All events in OMNeT are encapsulated as messages (simulation events and simulation-control events).
- Modules are in C++ and configuration in NED files and an .ini file.
- Various frameworks exist to simulate networks on the OMNeT platform. Two main frameworks:
  - INET (more for higher layer protocols and applications)
    - contains models for several wired and wireless networking protocols, including UDP, TCP, SCTP, IP, IPv6, Ethernet, PPP, 802.11, MPLS, OSPF
  - MiXiM (more for PHY and MAC simulation)
    - radio wave propagation, interference estimation, radio transceiver power consumption and wireless MAC protocols
OMNeT++ setup

- Connections between layers are done using “gates”.
- C++ files specify the behaviour of the modules
- The NED files specify how gates are connected.
- In the .ini file specifies parameters
  - Transmitter power, receiver sensitivity, thermal noise, slot durations, etc.
void TraCIDemo::initialize(int stage) {
    BaseApplLayer::initialize(stage);
    if (stage == 0) {
        debug = par("debug");
        traci = TraCIMobilityAccess().get(getParentModule());
        sentMessage = false;
        Move move;
        catMove = utility->subscribe(this, &move, findHost()->getId());
    }
}

void TraCIDemo::handleSelfMsg(cMessage *msg) {
}

void TraCIDemo::handleLowerMsg(cMessage* msg) {
    if (!sentMessage) sendMessage();
    delete msg;
}

void TraCIDemo::receiveBBItem(int category, const BBItem *details, int scopeModuleId) {
    BaseApplLayer::receiveBBItem(category, details, scopeModuleId);

    Enter_Method_Silent();

    if (category == catMove) {
        handlePositionUpdate();
    }
}

void TraCIDemo::sendMessage() {
    sentMessage = true;
    ApplPkt *pkt = new ApplPkt("BROADCAST_MESSAGE", 0);
    pkt->setDestAddr(-1);
    pkt->setSrcAddr(myApplAddr());
    pkt->setBitLength(headerLength);
    pkt->setControlInfo(new NetwControlInfo(L3BROADCAST));
    sendDown(pkt);
NED files (there are many)

In TraCIDI demo. NED (application layer)

```plaintext
simple TraCIDI demo like IBase ApplerLayer
{
    parameters:
    bool debug = default(false); // output debugging information
    int headerLength @unit("bit") = default(0bit); // length of the
    @display("i=block/app2");

    gates:
    input lowerGateIn; // from network layer
    output lowerGateOut; // to network layer
    input lowerControlIn; // control from network layer
    output lowerControlOut; // control to network layer
}
```

In Car. NED:

```plaintext
connections:
    nic.upperGateOut --> net.lowerGateIn;
    nic.upperGateIn <-- net.lowerGateOut;
    nic.upperControlOut --> { display("ls=red;m=m,70,0,70,0"); } --> net.lowerControlIn;
    nic.upperControlIn <-- { display("ls=red;m=m,70,0,70,0"); } <-- net.lowerControlOut;

    net.upperGateOut --> appl.lowerGateIn;
    net.upperGateIn <-- appl.lowerGateOut;
    net.upperControlOut --> { display("ls=red;m=m,70,0,70,0"); } --> appl.lowerControlIn;
    net.upperControlIn <-- { display("ls=red;m=m,70,0,70,0"); } <-- appl.lowerControlOut;
```
Initialization file

In config.ini:

```
*.[node*].nic.phy.usePropagationDelay = false
*.[node*].nic.phy.thermalNoise = -100dBm
*.[node*].nic.phy.useThermalNoise = true

*.[node*].nic.phy.analogueModels = xmlDoc("config.xml")
*.[node*].nic.phy.decider = xmlDoc("config.xml")

*.[node*].nic.phy.timeSleepToRX = 0.000102s
*.[node*].nic.phy.timeSleepToTX = 0.000203s

*.[node*].nic.phy.sensitivity = -80dBm
*.[node*].nic.phy.maxTXPower = 100.0mW
```

For each node, you can also specify location and speed, managed by the mobility module.

```
BaseNetwork.node[0].mobility.x = 150
BaseNetwork.node[0].mobility.y = 200
BaseNetwork.node[0].mobility.z = 250
BaseNetwork.node[*].mobility.speed = 1mps
```
OMNeT++ simple mobility demo

- BaseNetwork example that ships with MiXiM installation, just run the config.ini file (right click, run as OMNeT simulation)
Veins setup

- Veins connects SUMO and OMNeT++

- Veins uses a TCP connection and Python scripts to enable SUMO to act as a mobility model in OMNeT++

- Python set up to wait for Veins (module in OMNeT++)

```
sumo-launchd.py -p 9999 -vv -c /c/user/src/sumo/bin/sumo.exe
```

- And OMNeT is configured to look for mobility module

```javascript
# TraCIScenarioManager parameters
#

*.manager.updateInterval = 1s
*.manager.host = "localhost"
*.manager.port = 9999
*.manager.moduleType = "org.mixim.examples.traci_launchd.Car"
*.manager.launchConfig = "xml doc("sumo-launchd.launch.xml")"
*.node[*].applType = "TraCIDemo"
```
Veins (OMNeT++ & SUMO) demo

- Demo that ships with Veins installation, just run the config.ini file (Traci launch demo) (right click, run as OMNeT simulation)
- Only thing replaced are the SUMO net.net.xml and routes.rou.xml files.
Remaining challenges

- Data dissemination
- Signal propagation with obstacles
- Multichannel management in IEEE 802.11p
Questions or comments?