CISTER
Research Centre in
Real-Time Computing Systems

RoutesMobilityModel: Easy Realistic Mobility Simulation using External Information Services

Tiago Cerqueira, Michele Albano
CISTER Research Centre, ISEP, Portugal
www.cister.isep.ipp.pt

Workshop on Network Simulator 3, 2015
May 13th, 2015

isep Instituto Superior de Engenharia do Porto

Outline

- Mobility models for ns-3
  - ns3::RandomWaypointMobilityModel
  - SUMO
- Using external information services
  - Google Maps API
- ns3::RoutesMobilityModel
  - Features
  - Results
  - Internals
  - Limitations
- Future (current) work

Mobility Models for ns-3

- Simulation of mobile communication makes use of mobility models
- ns-3 possesses several synthetic mobility models implemented, such as:
  - Random Waypoint Mobility Model
  - Random Walk 2D
  - Gauss-Markov Mobility Model
- Another approach is coupling ns-3 with a traffic simulator, such as
  - Simulation of Urban MObility (SUMO)
  - Multi-agent Microscopic Traffic Simulator (MMTS)

ns3::RandomWaypointMobilityModel

- This module picks for each node a destination and velocity at random. When the node reaches the destination, it pauses for a specified amount of time and restarts the process.

- Easy to configure:
  - Only parameters: area where to place the nodes, speed range, pause time

- Disadvantages
  - Not much realism (constant speed, random positions, random destinations, does not consider the underlying road network)

SUMO

- SUMO is a microscopic vehicular traffic simulator
- Provides lots of interesting features
  - Can model vehicles, pedestrians and public transport
  - Can import maps, or generate custom road networks
  - Can model car following models and inner junction traffic
  - Gas emissions, traffic light frequency, etc
- Disadvantages
  - Steep learning curve
  - Can potentially take long time to configure since many details can/have to be specified
  - The user documentation consists mainly on its wiki
  - The SUMO architecture consists on a number of small programs that comprise the SUMO suite
    - Need to know, and possibly configure and run each tool to maximize realism

Using an external Information Service

- Synthetic mobility models in ns-3 are relatively simple and straightforward to use
  - However, they suffer from a low degree of realism.
- Full-fledged traffic simulators such as SUMO are more realistic
  - Time-consuming configuration
  - Easy to misconfigure / need expertise in vehicular traffic simulation
- We propose to distill data from travel planning services into mobility directives
  - We built a prototype that makes use of Google Maps API

Why Google Maps API?

- The Google Maps Web Services are one of the main core products of Google and provide geographic data for applications
  - Good support to developers
  - Feature rich

- We propose to use two APIs in particular:
  - Directions API
    - This API retrieves direction information between two locations
  - Places API
    - This API retrieves real world places (restaurants, hospitals, etc) around a given real world location

ns3::RoutesMobilityModel

- What it is
  - An interface to request directions between two (or more) points from external services, and parse them into a ns-3 mobility
  - A trade off between complex traffic simulations and easy (unrealistic) synthetic mobility trace generators
- What it isn't
  - A full-fledged traffic simulator for ns-3

- Currently, implemented for Google Maps only
- Can function online (contacting external service) or offline (importing travel plans from filesystem)
- Allows to select transportation method (driving, walking, cycling and public transportation)
- Allows to specify a departure time
  - Especially important when modeling public transportation

Visual comparison

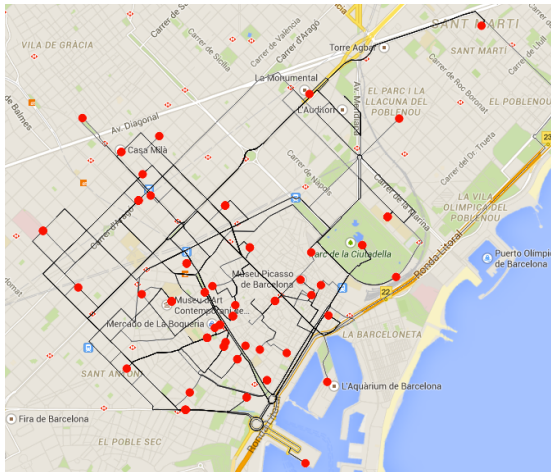


Figure 1 – RoutesMobilityModel

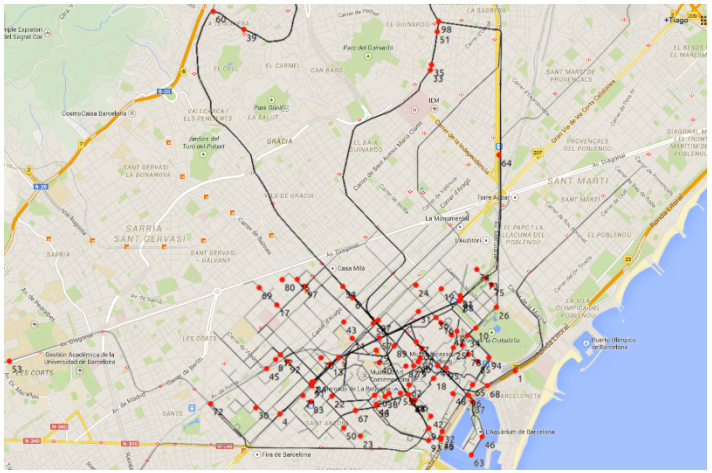


Figure 2 – SUMO

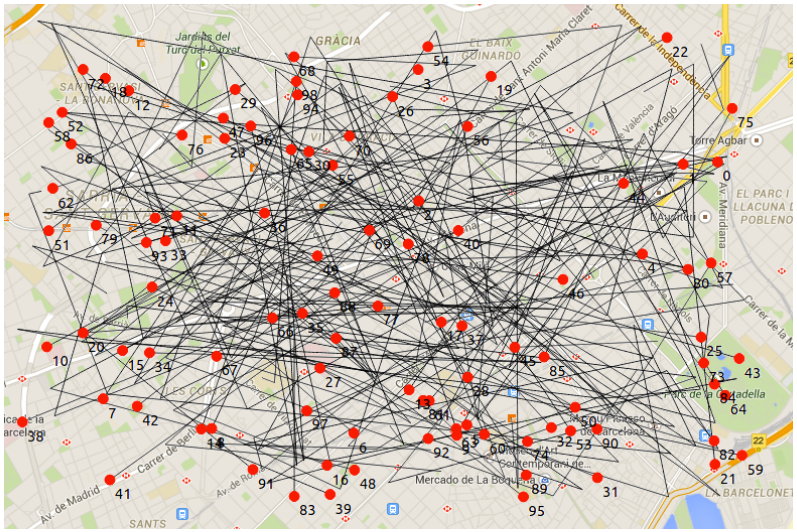


Figure 3 – RandomWaypointMobilityModel

Configuration complexity comparison

```
MobilityHelper mobility;
//Assign initial random positions
ObjectFactory pos;
int64_t streamIndex = 0;
pos.SetTypeId ("ns3::RandomBoxPositionAllocator");
pos.Set ("X", StringValue ("ns3::UniformRandomVariable[Min=0.0|Max=4600.0]"));
pos.Set ("Y", StringValue ("ns3::UniformRandomVariable[Min=0.0|Max=3000.0]"));
pos.Set ("Z", StringValue ("ns3::UniformRandomVariable[Min=1.0|Max=2.0]"));

Ptr<PositionAllocator> pAlloc = pos.Create ()->GetObject<PositionAllocator> ();
streamIndex += pAlloc->AssignStreams (streamIndex);
//Configure the speed and the pause time
std::stringstream ssSpeed;
ssSpeed << "ns3::UniformRandomVariable[Min=0.0|Max=" << 11.2 << "]";
std::stringstream ssPause;
ssPause << "ns3::ConstantRandomVariable[Constant=" << 5 << "]";
mobility.SetMobilityModel ("ns3::RandomWaypointMobilityModel",
                           "Speed", StringValue (ssSpeed.str ()),
                           "Pause", StringValue (ssPause.str ()),
                           "PositionAllocator", PointerValue (pAlloc));
mobility.SetPositionAllocator (pAlloc);
//Install the mobility to the nodes
mobility.Install (nodes);
streamIndex += mobility.AssignStreams (nodes, streamIndex);
```

Figure 4 – ns3::RandomWaypointMobilityModel configuration

```
MobilityHelper mobility;

mobility.SetMobilityModel ("ns3::WaypointMobilityModel");

mobility.Install(nodes);

RoutesMobilityHelper routes (41.306717, 2.119782,0);

routes.ChooseRoute(nodes,41.385329, 2.179434,2000);
```

Figure 5 – ns3::RoutesMobilityModel configuration

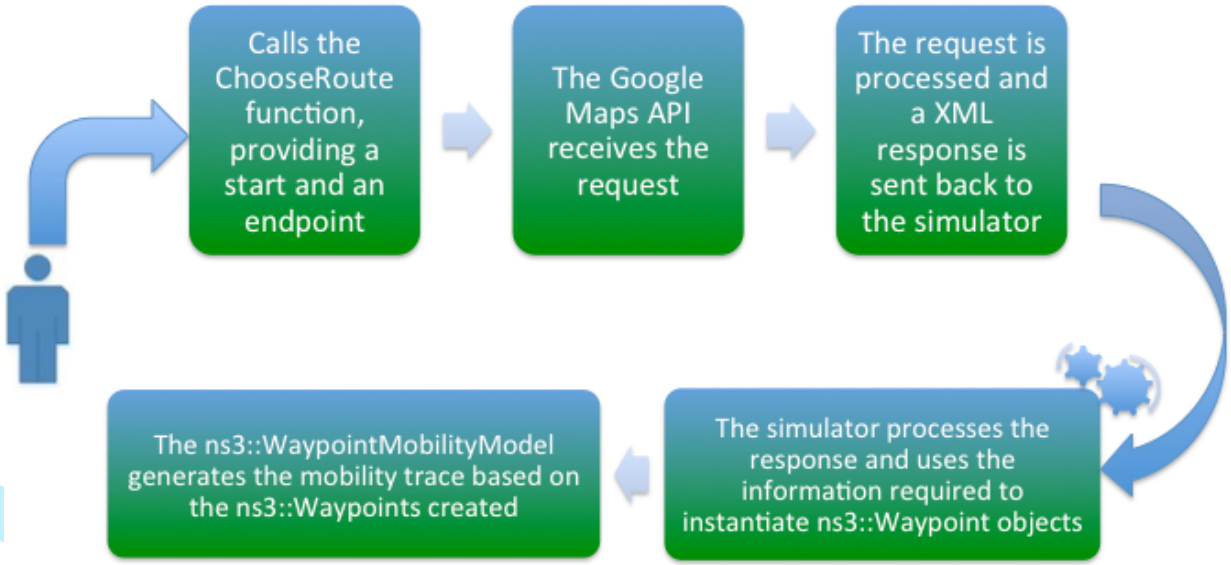
```
Export OSM map
netconvert --osm-files Downtown\ Barcelona.osm -o dwntwnBarcelona-net.xml
polyconvert --net-file dwntwnBarcelona-net.xml --osm-files Downtown\ Barcelona.osm --type-file typemap.xml -o DowntownBarcelona-poly.xml
python /usr/share/sumo/tools/trip/randomTrips.py -n dwntwnBarcelona-net.xml -r barcelona-rand.xml -e 50 –l
sumo -c config.sumo.cfg --fcd-output sumoTraceBarcelona.xml
python ../tools/traceExporter.py --orig-ids --fcd-input sumoTraceBarcelona.xml --ns2mobility-output mobility.tcl

Each line corresponds to an action, which has got its own configuration file
Some of the files (e.g. typemaps.xml, a basic one is 36 lines) MUST be filled up for each scenario
```

Figure 6 – SUMO configuration

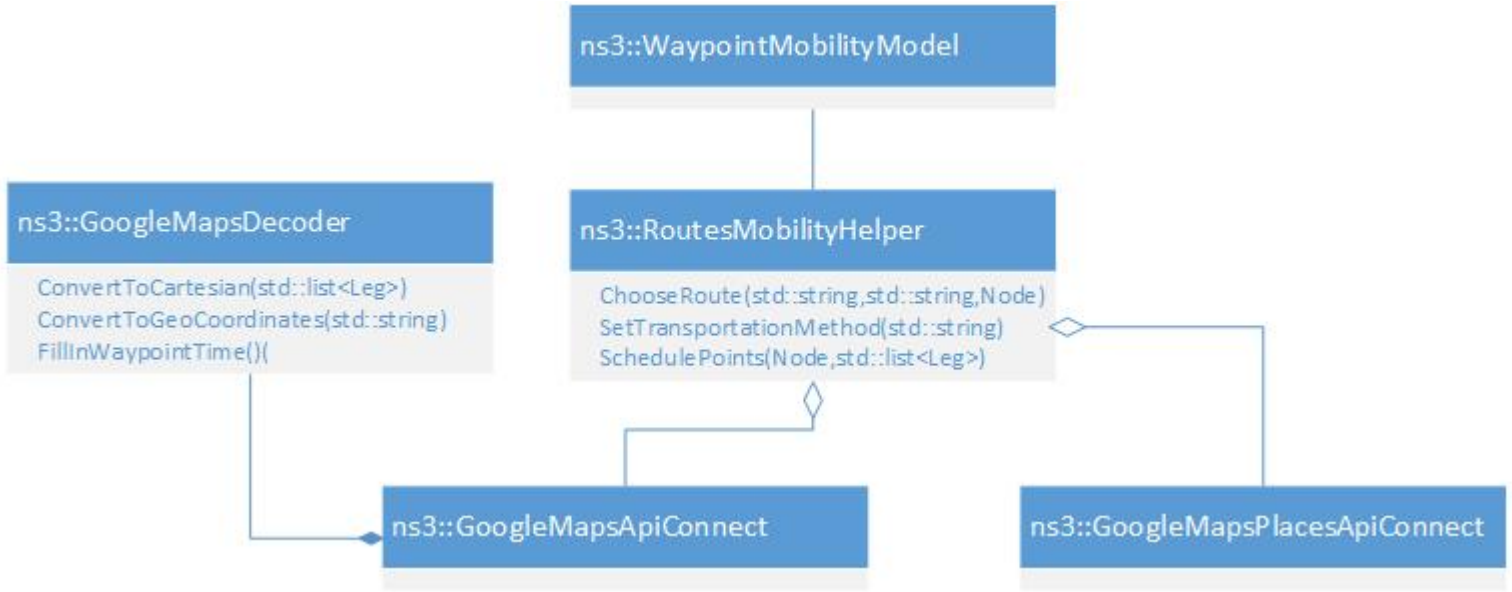
Internals 1/2

- The module was designed to import the routes via ns3::WaypointMobilityModel
  - Our module retrieves Geodetic points provided by Google Maps and converts them to Cartesian points, which can then be added to the WaypointMobilityModel



Internals 2/2

- The design of the module allows to download travel plans from different services, e.g. OSM
  - Module implemented using the Strategy Design Pattern, to adapt to additional external directions services
    - New strategy to access the service and parse the retrieved travel plan
  - Currently only the Google Maps services are being used



Limitations

- Current release of the module can/will be improved:
  - Only possible to generate mobility for node containers up to 30 nodes
    - Due to a limitation of the Places API
  - The module takes a long time to parse XML responses into mobility for large node containers
    - Still much faster than the execution of any non-trivial ns-3 simulation
  - No serialization is implemented
  - No bidirectional coupling with ns-3 is implemented
    - Currently, mobility impacts on what happens ICT-side
    - We want to allow ICT (car's computations) to lead to decision in changing car's routes
  - Traffic information is not available in the free version of the API

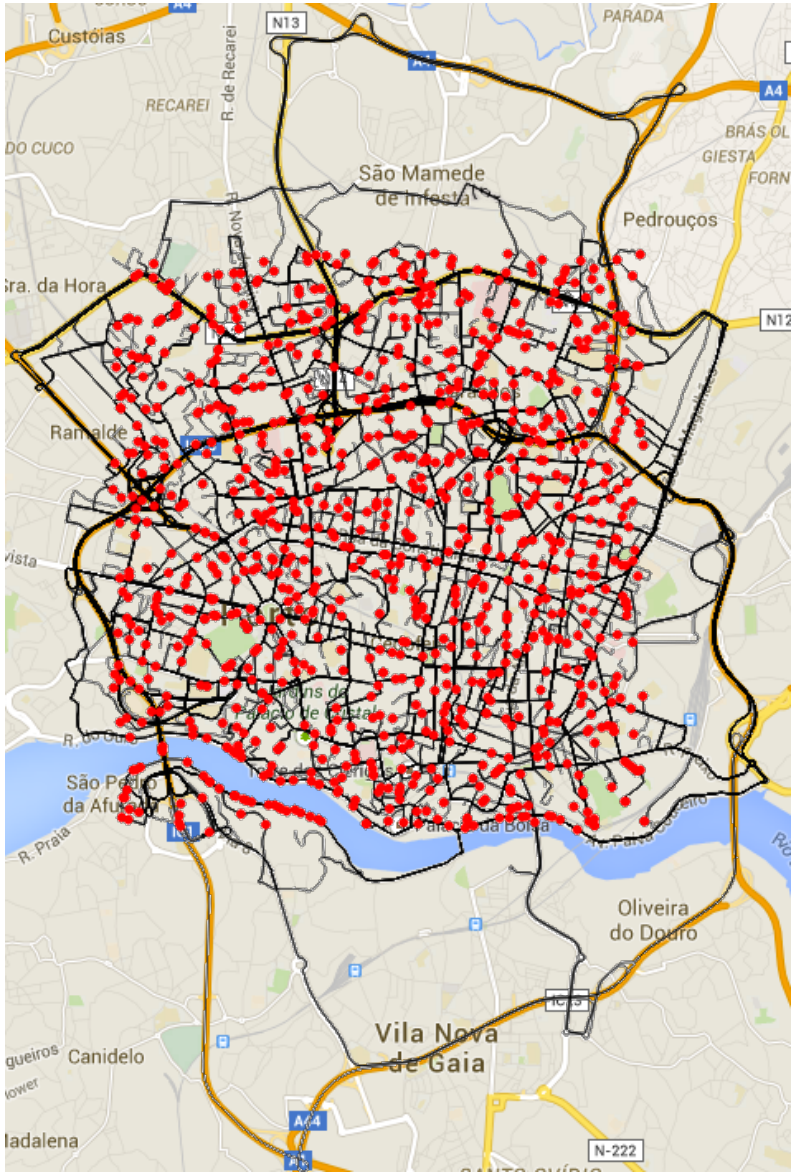
Future/current work

- Removing the 30 node limitation
- Switch from XML to JSON
  - Potential of speeding up the parsing of mobility data
- Implement serialization
  - To allow the user to repeat the simulation using the exact same mobility
- Bidirectional coupling with the simulator
- Thorough evaluation of the module
  - Other metrics
  - Other mobility models

Sneak preview of the next release

- Already available in the repository
  - NOT under review for ns-3 inclusion
  - Will not be included in ns-3's next version
    - Hopefully will be included later on
- The 30 node limitation was removed
  - By combining the 60 places returned by the Places API at random
  - By combining user specified locations at random
  - By randomly choose coordinates in a user specified location
- Serialization
  - In the current release if a user manually downloads a travel plan, the module can already parse it as mobility

Sneak preview of the next release



Thank you for your attention

- Current testing repository
  - https://bitbucket.org/TiagoCerqueira/routesmobilitymodel
- Wiki
  - https://www.nsnam.org/wiki/RoutesMobilityModel
- Code review issue
  - https://codereview.appspot.com/176430044/

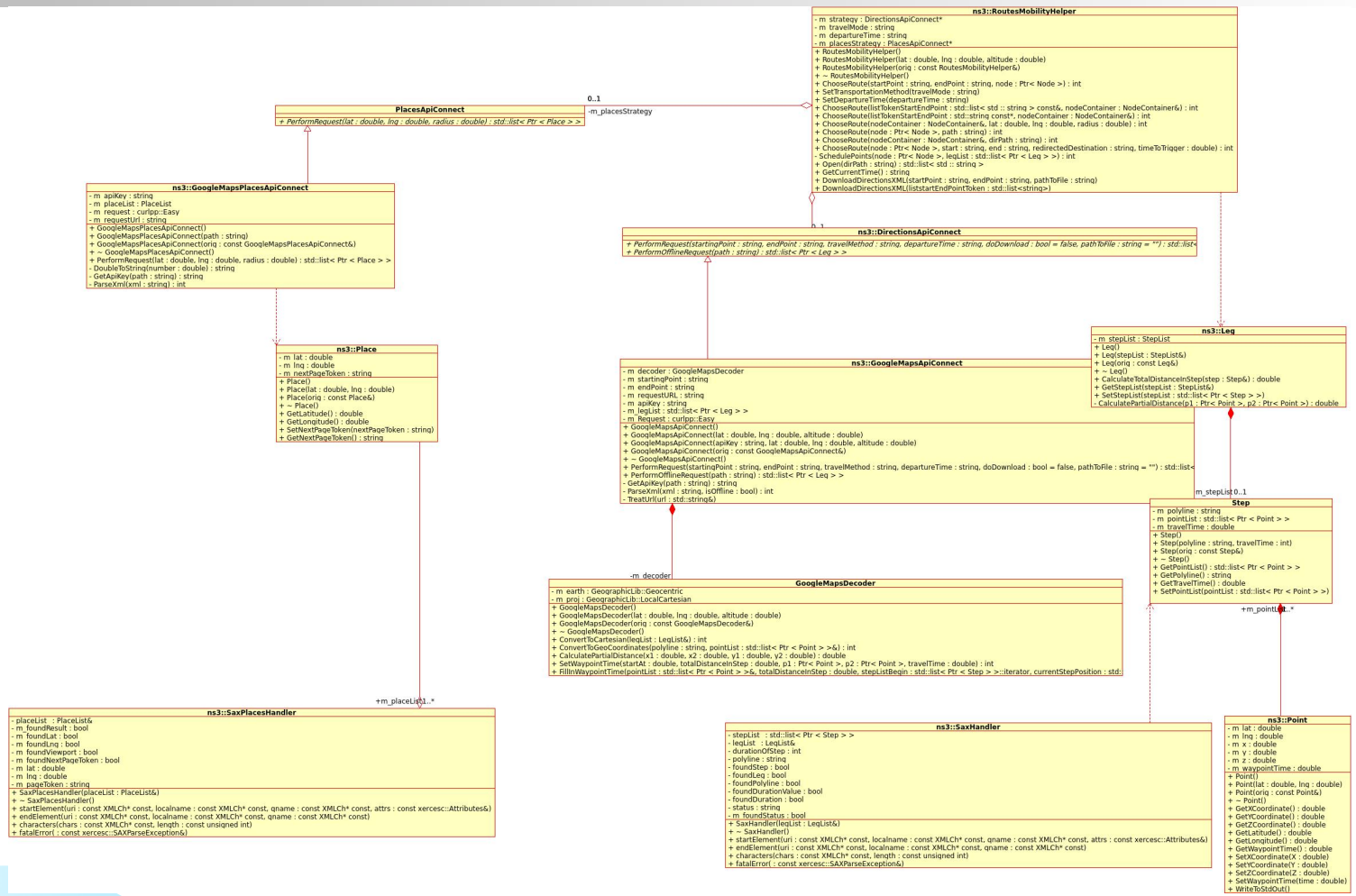
More material

- Both these APIs are free and have usage quotas
  - Directions API
    - 2.500 Directions requests per 24 hour period
    - 2 requests per second
    - No traffic information is available on the free version
  - Places API
    - 1000 requests per 24 hour period, which can be increased to 150 000 requests by verifying the user's identity with a credit card.
    - Unfortunately it's only able to return up to 60 places in any query (despite the fact that there are more locations in that area)

More material

- Caching the API's responses for more than 30 days is a direct violation of the ToS. Because of this, no caching feature was implemented. However, it is possible to load XML responses from the filesystem

More material



MacPhy overhead comparison

- An analysis of the performance of the routing protocol AODV was performed using vanet-routing-compare script, in order to further validate the results
  - Three scenarios where tested:
    - A scenario using the ns3::RoutesMobilityModel on Downtown Barcelona
    - A scenario using SUMO on Downtown Barcelona
    - A scenario using the ns3::RandomWaypointMobilityModel using the same area as the area in the scenarios using SUMO and the RoutesMobilityModel.
- The simulations where ran using 99 nodes, for 300 simulated seconds. The vehicles randomly chose the route to take.