CISTER
Research Center in
Real-Time & Embedded
Computing Systems

Technical Report

RoutesMobilityModel: easy realistic mobility simulation using external information services

Tiago Cerqueira
Michele Albano

CISTER-TR-150302

2015/05/13

RoutesMobilityModel: easy realistic mobility simulation using external information services

Tiago Cerqueira, Michele Albano

CISTER Research Center

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8321159

E-mail: 1090678@isep.ipp.pt, mialb@isep.ipp.pt

http://www.cister.isep.ipp.pt

Abstract

The current implementation of ns-3 provides only synthetic mobility models that disregard the map where the nodes are moving, however, the study of vehicular ad-hoc networks requires the usage of more realistic mobility models. The usage of mobility traces created by traffic simulators such as SUMO is feasible, however, these simulators possess a steep learning curve, which impedes their fruition for most researchers whose research focus and expertise are on the data communication layer.

This paper presents a mobility model that generates realistic mobility traces that take into account the underlying maps, while maintaining the ease of usage that characterizes the synthetic mobility models. The module described herein is compared against SUMO and against the ns3::RandomWaypointMobilityModel of network simulator 3, to analyze the trade-off it implements in terms of realism and ease of usage.

RoutesMobilityModel: Easy Realistic Mobility Simulation using External Information Services

Tiago Cerqueira
CISTER, ISEP/INESC-TEC
Rua Dr. António Bernardino de Almeida 431
4249-015, Porto, Portugal
1090678@isep.ipp.pt

Michele Albano
CISTER, ISEP/INESC-TEC
Rua Dr. António Bernardino de Almeida 431
4249-015, Porto, Portugal
mialb@isep.ipp.pt

ABSTRACT

The current implementation of ns-3 provides only synthetic mobility models that disregard the map where the nodes are moving, however, the study of vehicular ad-hoc networks requires the usage of more realistic mobility models. The usage of mobility traces created by traffic simulators such as SUMO is feasible, however, these simulators possess a steep learning curve, which prevents their fruition for most researchers whose research focus and expertise are on the data communication layer.

This paper presents a mobility model that generates realistic mobility traces that take into account the underlying maps, while maintaining the ease of usage that characterizes the synthetic mobility models. The module described herein is compared against SUMO and against the `ns3::RandomWaypointMobilityModel` of network simulator 3, to analyze the trade-off it implements in terms of realism and ease of usage.

Categories and Subject Descriptors

C.2.2 [**Network Protocols**]; I.6.5 [**Model development**]; I.6.7 [**Simulation support system**]

General Terms

Algorithms, Design, Experimentation

Keywords

Mobility model, trade-off analysis, network simulator 3

1. INTRODUCTION

Research and development activities in the fields of communication networks traditionally leverage simulations to explore the most probable results of the deployment of a given solution in a real scenario. Mobile communication is not an exception, and a plethora of simulators [1] were developed to cope with the problem of performing realistic simulations of mobile systems that communicate with each other and with a fixed infrastructure.

The research area of *Vehicular Ad-Hoc Networks* (VANETs) has seen accelerated development in recent years since the academia and the industry have produced specialized protocols [2], which have the potential to empower *vehicle-to-vehicle* (V2V) and *vehicle-to-infrastructure* (V2I) scenarios with communication capabilities, including safety related applications and traffic and fleet management. To satisfy research and development needs, a number of simulators have been either developed or adapted for vehicular communication simulation [3]. In order to correctly model these networks in a network simulator, a realistic mobility model must be used, as the mobility patterns have a significant impact on the vehicle's ability to communicate [4]. Current results have shown that synthetic mobility models, computed without taking into account the underlying road networks, can hardly be tailored to real maps and scenarios [5].

A few proposals have created vehicular traffic simulators [6], which are much more realistic and can be tailored to real-world maps. However, these simulators are quite complex to set up and need a solid experience in vehicular traffic simulation. Moreover, these traffic simulators are unable to simulate the effect of in-vehicle applications, which alter the vehicle's behaviour, unless Application Programming Interfaces (APIs) such as TraCI are used, adding yet another layer of complexity [7, 8].

This paper provides the description of a mobility model called RoutesMobilityModel that performs a trade-off between the synthetic mobility models, and complex vehicular traffic simulators. Our system accepts as inputs the details of the trajectory at large, connects to a travel planning service to download directions from the source to the destination of the planned trip, and converts it to a mobility trace for the simulator at hand. In particular, we implemented a prototype that makes use of Google Maps Service [9] in order to create mobility traces for the network simulator 3 (ns-3) [10].

The rest of the paper is structured as follows. Section 2 provides insights into existing mobility models, and the technologies we built upon while developing RoutesMobilityModel. Section 3 discusses the motivations that led us to develop RoutesMobilityModel. Section 4 describes the design and implementation of the module, Section 5 evaluates our module and the trade-off it implements between synthetic approaches and complex realistic approaches. Finally, Section 6 discusses limitations of the module and future work,

WNS3 2015, May 13 2015, Barcelona, Spain

DOI: http://dx.doi.org/10.1145/2756509.2756515.

and Section 7 wraps up the paper.

2. RELATED WORK

This sections starts with a brief description of existing mobility models for vehicular networks, afterwards it describes the existing technologies that we built upon and integrated to design and implement the RoutesMobilityModel.

2.1 Existing Approaches

Currently, vehicular mobility is simulated either by using complex traffic simulators, such as Simulation of Urban Mobility (SUMO), or using synthetic mobility models, such as the random waypoint model, the random walk model or the Gauss-Markov model [3].

SUMO provides the most realistic mobility traces, however its higher complexity with respect to other mobility models (see for example [5]) makes it a burden to configure when the researcher's area of interest and expertise is on the data communication only. SUMO is capable of generating predefined abstract road networks or importing a digital road from services such as OpenStreetMaps or other traffic simulators such as VISUM, Vissim or MATSim. Creating a real-world road topology, however, is still a time consuming task, as configuration of complex scenarios, such as intersections, number of lanes, right of way, etc, still require a great deal of work [11].

Random mobility models, such as the random waypoint model and the Gauss-Markov mobility model are able to provide easy and fast mobility for network simulators, but their nature can often lead to unexpected results. In fact, these systems do not take the underlying maps into account, thus the realism of the mobility traces they generate is limited, and the importance of developing mobility models that take into account the underlying maps are testified by the multiple research activities in this area [3, 4, 5, 6].

The random waypoint model works by having every node in the simulation pick a random destination and a random velocity in certain parts of the trajectory. This, in turn, means that the nodes will travel at a constant velocity between the two parts of the trajectory (waypoints) [12]. Upon arrival at a waypoint, the node will pause for a determined amount of time and restart the process.

The Gauss-Markov model assigns speed and direction to a given node, and generates mobility by updating the node's speed and direction at a specified interval of time. This module is an improvement over other synthetic modules as it allows past velocities and directions to influence the future velocities and directions [13].

2.2 The Google Maps API

Our mobility model is based on the information made available by the Google Maps suite of services [9]. This subsection provides a brief description of the API to interact with the two services that are most interesting for our work, the Google Maps Direction API and the Google Maps Places API.

2.2.1 The Google Maps Directions API

The Google Maps Directions API offers a way for developers to interact with the Google Maps Directions Service, providing travel planning information based on transportation method. Our module makes use of this API in order to request from Google the path between two real world locations. Encoded polyline strings are returned by the service, representing movement between geographical coordinates (latitude and longitude) [14]. The API's response is articulated as follows:

- Leg – A travel is composed by Legs. Legs only occur if the user specifies a Waypoint (A to B, passing through C, for example), and each Leg is the trajectory to reach one Waypoint
- Step – A Leg is composed by Steps. A Step contains a polyline string, as well as the time estimation for the user to go from the first point of the polyline string to the last one, based on the transportation method chosen.

2.2.2 The Google Maps Places API

The Google Maps Places API returns information regarding particular establishments or point of interests, such as hospitals, gas stations, shops, etc. This information is used in tandem with the Google Maps Directions API, retrieving locations from a specified area, which are then used as start and endpoints in queries to the Directions API. This enables the user to quickly and effortlessly generate mobility for a node container, without having to manually specify start and end points.

The API response consists of a location (latitude and longitude) and information about the Place (such as its rating, its type, etc). The RoutesMobilityModel module uses the location attribute only.

2.3 The ns3::WaypointMobilityModel

Our mobility model produces traces that are imported into the ns-3 through its `ns3::WaypointMobilityModel` module. This subsection describes this module, to lay out the basis for the interaction of RoutesMobilityModel with ns-3.

The `ns3::WaypointMobilityModel` is capable of providing waypoint-based mobility to a node. Each object determines its velocity and position at a given time from a set of `ns3::Waypoint` objects. These objects are stored in a double ended queue, and are discarded when the current simulation time is greater than the time value of the object. When a node is in between waypoint times, it moves at a constant speed between the current and the previous waypoint [12].

The usage of this mobility module requires users to provide the Cartesian coordinates and the time of passage for each `ns3::Waypoint`, which is a cumbersome task to generate manually even for a simple mobility trace. On the other hand, `ns3::RoutesMobilityModel` generates these data programmatically, based on the XML traces returned by the Google Maps API.

2.4 External Libraries

A number of external libraries were integrated into the design of RoutesMobilityModel, to enable a faster and more robust fruition of the data returned by the Google Maps API.

2.4.1 Xerces-C++

Xerces-C++ [15] is a validating XML parser written in a portable subset of C++. This library is used in conjunction with the module described in order to parse the responses from the Google Maps APIs. The module uses the SAX2 parser, which is an event-driven mechanism for accessing XML documents.

2.4.2 GeographicLib

GeographicLib is a small set of C++ classes for performing conversions between geographic, UTM, UPS, MGRS, geocentric and local Cartesian coordinates, for gravity, geoid height and geomagnetic field calculations. It is also used for solving geodesic problems [16]. The module makes use of the algorithms contained in GeographicLib library in order to, reliably, convert between World Geodetic System (WGS 84) coordinates and Cartesian coordinates.

2.4.3 libcurl

libcurl [17] is a free and easy-to-use client-side URL transfer library. The library supports, among others, the HTTPS protocol, which is required by the Google Maps APIs. This library is used to query the Google Maps APIs, via a regular HTTP request.

3. MOTIVATION

Simulations regarding Mobile Ad-hoc Networks (MANETs) and Vehicular Ad-hoc networks (VANETs) raise the need for proper mobility models. Many mobility models available in network simulators provide synthetic traces that disregard the maps the nodes are moving onto (see for example Random Walk 2D model, the Random Waypoint Model and the Gauss-Markov model). Research has shown that some aspects of vehicular traffic, such as acceleration and deceleration in the presence of obstacles, greatly affect the network performance [18]. Using traffic simulators such as SUMO within network simulators add realism, but a first time user with no experience with traffic simulators will encounter a steep learning curve. In order to generate a real-world road network, a SUMO user is required to run a handful of complex scripts, which require several configuration parameters each. The tutorial section in [11] provides a good reference to generate mobility traces ranging from the most basic to the most complex, however, it is quite lengthy and a user interested, mainly, in data communication is not likely to require such a high level of configuration.

The main goal of this paper is to describe a mobility model that is as easy to configure as the synthetic mobility models included within ns-3, while maintaining a high degree of realism, comparable to those offered by SUMO and similar software suites. Our module implements a trade-off between the two families of mobility models in terms of ease of usage and realism.

4. DESIGN AND IMPLEMENTATION

The module described in this paper provides an interface to convert information from a travel planning service to `ns3::Waypoints`, which are then used by ns-3 to manage the mobility of the simulated nodes. Currently, the only service accessible by `ns3::RoutesMobilityModel` is the Google Maps service [9]. The rationale behind this choice is that Google Maps service provides a robust and feature-rich API, which for example allows to select the mode of travel (walk, car, public transport). However, the design of our module allows easy addition of other services.

In order for the module to be useful for as many researchers as possible, the following use cases where taken into consideration while developing the module:

- Generation of mobility traces for a node, using either addresses or coordinates as start and end points
- Generation of mobility traces for all nodes contained into a `ns3::NodeContainer`
- Generation of mobility traces from previously downloaded responses
- Generation of mobility traces with dynamic node redirection

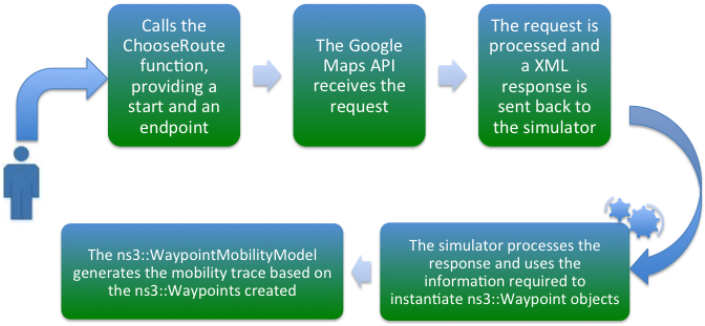

Figure 1: Module interaction when generating mobility traces

Our module uses the Google Maps Directions API that, given a start and an end point, computes the best path between them. The response of the API is then parsed by our module, in order to translate the response and create `ns3::Waypoints`. These waypoints are then imported in ns-3 through the `ns3::WaypointMobilityModel`, which processes the `ns3::Waypoints` and generates the corresponding mobility trace.

The usual interaction between the module, the simulator and the external information service is represented in Figure 1. The simulator invokes the `ns3::ChooseRoute` method, providing a start and an end point for the route. The RoutesMobilityModule contacts the Google Maps service, which answers with an XML file containing the route. The RoutesMobilityModule parses received data to create the list of waypoints corresponding to the route. Finally, the ns-3 simulator imports the waypoints to use them as the mobility pattern through its `ns3::WaypointMobilityModel` methods.

4.1 Module Architecture

The module described in this paper is articulated into several classes, which are responsible for the features described earlier in this section. The architecture is represented in Figure 2. Some of the most relevant classes include:

- RoutesMobilityHelper – helper class through which all of the features described in the paper are made available to the user. This class is responsible for creating `ns3::Waypoints` and adding them to the queue contained into the `ns3::WaypointMobilityModel` for further processing.
- GoogleMapsDecoder – responsible for decoding the polyline strings obtained from the Google Maps Directions API, into latitude and longitude pairs.
- GoogleMapsApiConnect and GoogleMapsPlacesApiConnect - responsible for querying the respective Google Maps APIs invoking the respective parsers for the retrieved XML files.

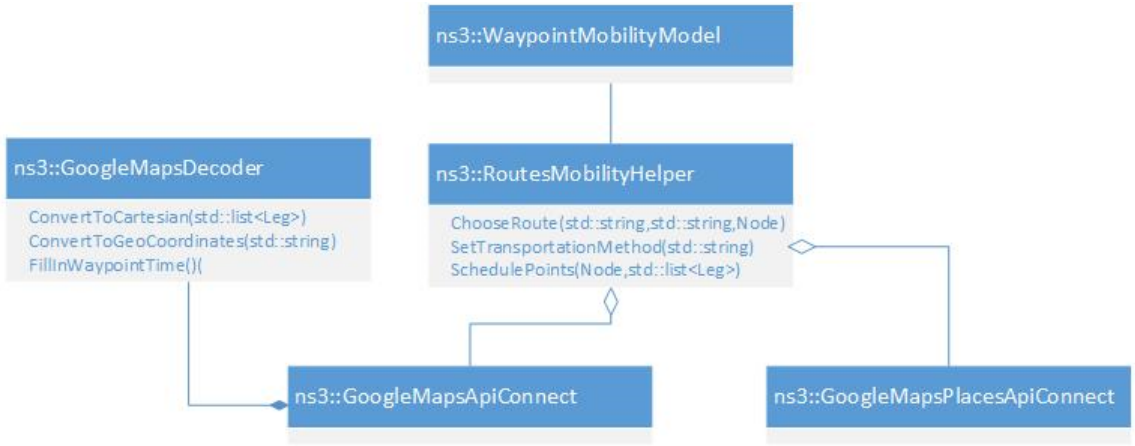

Figure 2: Architecture of the RoutesMobilityModel module

- SaxPlacesHandler and SaxHandler – responsible for parsing the Google Maps API's response.

The module was built in order to accommodate different travel planning services, such as OpenStreetMaps, and locations databases. To this end, the Strategy software design pattern was used in the module design. The Strategy pattern decouples the class' code from the algorithms it uses. This allows for the `ns3::RoutesMobilityHelper` to choose, at runtime, which travel planning service and location database to use.

The Strategy design pattern is also instrumental for easing up the implementation of new services that provide direction information or locations. In fact, the only required effort on part of the developer is the implementation of the provided interfaces and the usage of the provided model classes (`ns3::Leg`, `ns3::Step`, `ns3::Point` and `ns3::Place`) to represent the information retrieved by the implemented service.

4.2 Implementation

The RoutesMobilityModel module relies on the module `ns3::WaypointMobilityModel` in order to import the mobility routes retrieved from the external information services. Our module creates `ns3::Waypoints` that model the routes used to travel between two (or more) real world locations, and later on they are interpreted as the waypoints the vehicles move through during the simulation.

The information retrieved from the Google Maps Directions API contains, among other information, a polyline for each step of the route requested. It also contains the duration (the time it would take to go from the beginning to the end) for the step. The module decodes the polyline, thus creating a list of geographical coordinates, which are, in turn, converted to Cartesian coordinates. In order to model the speed of a node, implemented by setting the times for the `ns3::Waypoints`, we distributed the duration of the step in proportion to the distance traveled between two waypoints.

Three examples were also implemented, to provide users with additional information on the module and its usage:

- routes-mobility-example.cc – This example queries the Google Maps Directions API to generate a mobility trace based on a real-world route. The mobility traces generated in this example are from the city of Porto, Portugal
- routes-mobility-offline-example.cc – This example generates mobility traces based on Google Maps Directions API's XML files located on the hard disk. Before using the example, the user needs to download the file manually, as the Google Maps APIs Terms of Service forbid an application from caching the responses.
- routes-mobility-automatic-example.cc – This example uses the Google Maps Places API to generate mobility traces for node containers of up to 30 nodes. Mobility generation is done by querying the Google Maps Places API for places (restaurants, cinemas, etc), which will be used as start and end points. In this example, the area where the place are located is the city of Porto, Portugal.

5. VALIDATION

In order to validate the proposed mobility model, three scenarios were simulated, all of them featuring vehicles executing the same communication protocol between each other (the AODV routing protocol [18]), but with mobility generated using SUMO, the model `ns3::RoutesMobilityModel` and the model `ns3::RandomWaypointMobilityModel` respectively. This test leverages the test scripts that are part of the standard ns-3 distribution. Common parameters to the simulations were:

- 99 nodes
- 300 simulated seconds
- Vehicles randomly choose the route they take
- Nodes broadcast safety messages 10 times per second at 6 Mbps
- The TwoRayGround propagation model was used

A few parameters had to be different in the simulations, to take care of the structural diversity of the mobility models. To this end, the vanet-routing-compare.cc script was modified to simulate the following scenarios:

- SUMO scenario:

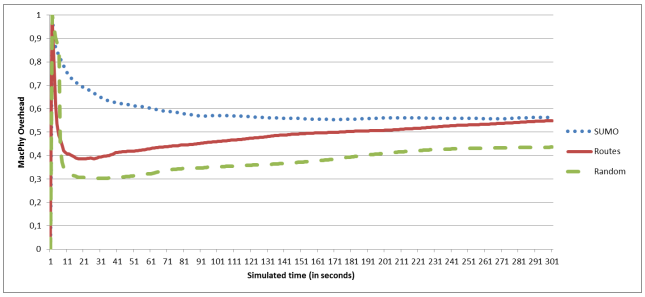

Figure 3: Overhead of the communication

  - Mobility generated using SUMO for an area in downtown Barcelona, with width 4.6 km and height 3.0 km.

- RoutesMobility scenario:
  - Mobility in an area in downtown Barcelona, generated using the `ns3::RoutesMobilityModel`. Start and end points are obtained through a query to the Google Maps Places API. The radius requested was of 2.1 km, which leads to roughly the same area as the other scenarios.
- RandomWaypoint scenario:
  - Mobility generated in an area 4.6 km x 3.0 km using the `ns3::RandomWaypointMobilityModel`. We can consider the area to be located in downtown Barcelona.

The configuration of the SUMO scenario took a few hours, on the other hand both the `ns3::RoutesMobility` scenario and `ns3::RandomWaypoint` scenario were quite straightforward. In particular, RoutesMobilityModel required only the specifying of the kind of places to be retrieved, the center of the simulation area, and its radius.

The simulation results evaluated the communication overhead in terms of the ratio between the total bits sent on the wireless medium, and the payload net of MAC and PHY overhead. We conjectured that our mobility model led to AODV communication to presenting a performance similar to the case of SUMO mobility, while both being quite different from `ns3::RandomWaypointMobilityModel`. Simulation results, reported in Figure 3, correspond to the mean behavior of the AODV protocol over 10 simulations. The results confirm the convergence over time in the behavior of SUMO and `ns3::RoutesMobilityModel` and their difference from `ns3::RandomWaypointMobilityModel`, thus providing a hint regarding the relative realism of our module with respect to `ns3::RandomWaypointMobilityModel`.

The mobility traces generated are also quite similar in the SUMO scenario and in the RoutesMobility scenario. Figure 4 and Figure 5 show the mobility routes generated in the SUMO scenario and in the RoutesMobility scenario respectively, with the bullets represent initial location of vehicles, and the lines represent the routes taken by the vehicles according to the mobility model.

The mobility generated in the RandomWaypoint scenario, shown in Figure 6, is drastically different from the two mobility traces presented above.

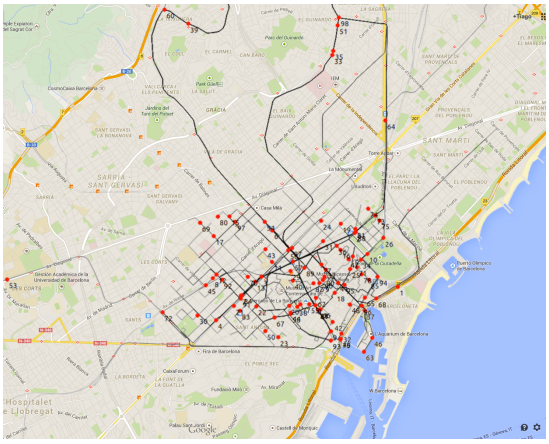

Figure 4: Mobility of SUMO scenario

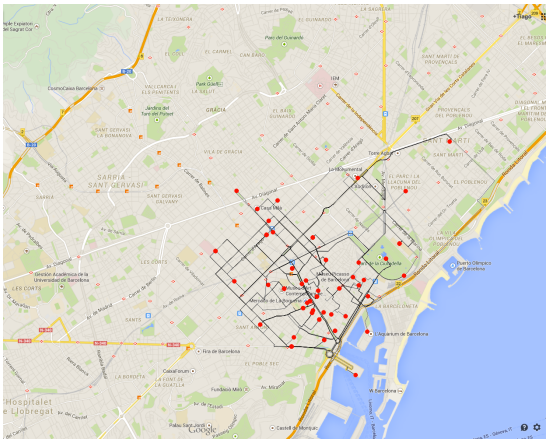

Figure 5: Mobility of RoutesMobility scenario

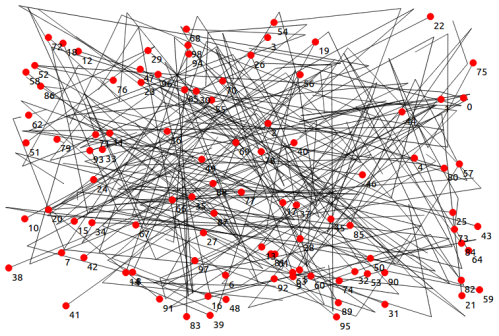

Figure 6: Mobility of RandomWaypoint scenario

The results shown in this section hint at the usefulness of the RoutesMobilityModel module to researchers studying VANETs or MANETs, as the module implements a nice trade-off between ease of usage and realism.

The mobility generated using the module described in this paper is accurate both in space and time, as it uses

the partial trip time, as calculated by the Google Maps Directions API, to decide at which time it places a specific `ns3::Waypoint`. This, in practice, means that the nodes will always travel the speed Google expects a real car to travel through that particular road, taking into account kind of road, presence of roundabouts, sharp turns, onramps, etc. Moreover, if traffic information is available for the route chosen, the car will travel the sections under traffic at a realistic speed. This feature is only available using the "Google Maps APIs for Work" service, which is a pay service, otherwise traffic information is not included.

The RouteMobilityModel module is currently being used in simulative analysis of data dissemination algorithms in vehicular environments.

6. LIMITATIONS AND FUTURE WORK

Our current research work aims at developing further the module, since it still possesses room for improvement.

Firstly, the module will only be useful if it is able to provide mobility for a node container of a sufficient size. Currently, the Google Maps Places API is only able to return up to 60 locations, which greatly impairs this module. Because of this, the current implementation only supports automatic mobility generation for node containers of 30 nodes, however, several solutions for this problem have been proposed:

- Randomly choose a start and an end point from the places returned by the query to the Google Maps Places API.
- Use a different service which is capable of returning more than 60 locations.
- Generate routes by combining pairs of points taken from the same set, producing a number of routes that scale as the square of the number of points. A drawback of this approach is that the nodes would essentially keep traveling using many times the same streets / freeways, etc.
- Generate routes by randomly select a start and endpoint from a user-specified pool of locations
- Generate routes by combining pairs of points, the first one being chosen at random in a set of starts points, the second one chosen at random in a set of destination points. Similar to the previous one, this solutions scales quadratically as the number of points in the two sets.

The first proposed solution was tested successfully. The resulting mobility appears to be realistic with respect to typical behaviors of car drivers.

In an experimental version of the RoutesMobilityModel module, the fourth and fifth solutions were also tested. By defining manually the sets of points, it is possible to provide mobility for containers of any size. By using this technique in conjunction with the Google Places API, using the latter to create a pool of address to use as start and end points, it is possible to automatize the selection of points and still generate a large number of routes. The approaches mentioned in this paragraph were tested with positive results with 240 nodes (Figure 7 and Figure 8).

The computational time of the module is another drawback that must be addressed in the future. Currently, the

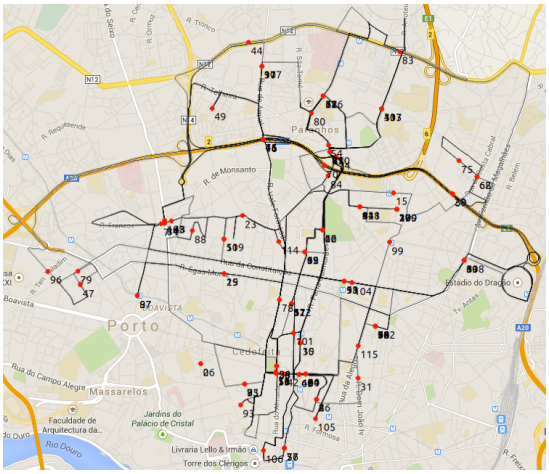

Figure 7: Mobility generated using user specified locations

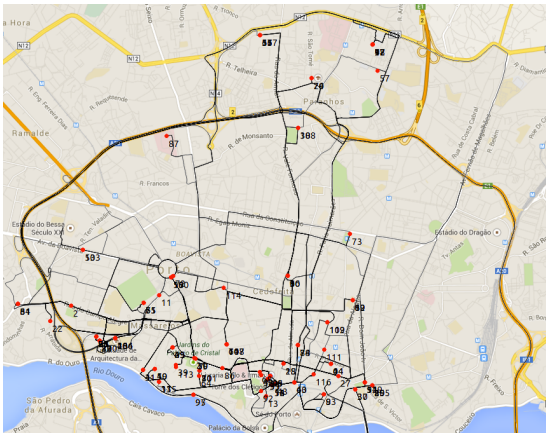

Figure 8: Mobility generated using the Google Maps Places API

module takes a long time to generate mobility for large node containers, since it is required to parse a large XML response per node in the container. The switch from an XML parser to a JSON parser can provide valuable performance enhancement, and it is a planned future work.

Currently, the module is also lacking proper serialization capabilities, which is the capability of the module of storing on the filesystem the data returned by the external information service, with the aim of parsing the data to a ns-3 mobility model at a later time and potentially multiple times to guarantee repeteability of the simulation. A few efforts were made in the early stages of the development, however, no serialization is implemented at this time. This matter should be studied and implemented in the future, as proper serialization is key, both to improve the usability of the module by decreasing the computational time for creating the `ns3::Waypoints`, and to allow users to generate a simulation scenario once and load it as needed.

The module also lacks proper bidirectional coupling between the network simulator's network modules and the mobility model created. That is, the module does not react to

the data a node receives. It should be possible for the users to specify actions such as redirecting a node based on data received, for example. We plan to develop an API to allow for this bidirectional coupling, empowering users to enable the nodes to make traffic routing decisions based on data received.

A last improvement we propose for our module aims at exploiting the data from travel planning services and location databases different from the Google Maps services.

Finally, regarding the investigation methodology, we have planned to perform a more proper comparison with other mobility models, both synthetic such as the Gauss-Markov model and realistic such as the multi-agent microscopic traffic simulator (MMTS) [19], and consider more metrics to evaluate the realism and ease of usage of our mobility model.

7. CONCLUSION

The work presented in this paper details a mobility model capable of generating mobility with an acceptable degree of realism, while maintaining the same (or even greater) ease of use that characterize the synthetic mobility models that do not take into account real-world maps. The work described was validated through simulation scenarios, with encouraging results. Results showed that, while RoutesMobilityModel is not as realistic as traffic simulators, it still maintains an acceptable level of realism for VANETs. MANET researchers will also be able to take advantage of this module, since Google Maps API allows to model movement for nodes traveling on foot or using public transportation.

8. ACKNOWLEDGMENTS

This work was supported by the Portuguese Agency for Innovation (ADI) under the ERDF (European Regional Development Fund) through COMPETE (Operational Programme 'Thematic Factors of Competitiveness'), within project CARCODE, ITEA2 Nr. 11037, QREN - SI I&DT Nr. 30345.

9. REFERENCES

[1] E. Weingartner, H. Vom Lehn, and K. Wehrle. "A performance comparison of recent network simulators." IEEE International Conference on Communications (ICC'09), 2009.

[2] Y.-A. Daraghmi, C.-W. Yi, and I. Stojmenovic. "Forwarding methods in data dissemination and routing protocols for vehicular ad hoc networks." Network, IEEE 27.6, pp. 74-79, 2013.

[3] F. J. Ros, J. A. Martinez, and P. M. Ruiz. "A survey on modeling and simulation of vehicular networks: Communications, mobility, and tools." Computer Communications 43, pp. 1-15, 2014.

[4] F. K. Karnadi, Z. H. Mo, and K. Lan. "Rapid generation of realistic mobility models for VANET." IEEE Wireless Communications and Networking Conference (WCNC 2007), 2007.

[5] F. J. Martinez, C. K. Toh, J. C. Cano, C. T. Calafate, and P. Manzoni. "A survey and comparative study of simulators for vehicular ad hoc networks (VANETs)." Wireless Communications and Mobile Computing 11.7, pp. 813-828, 2011.

[6] V. D. Khairnar and S. N. Pradhan. "Comparative study of simulation for vehicular ad-hoc network." arXiv preprint arXiv:1304.5181, 2013.

[7] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker. "Recent development and applications of SUMO – simulation of urban mobility." International Journal on Advances in Systems and Measurements 5.3 and 4, pp. 128-138, 2012.

[8] C. Sommer, Z. Yao, R. German, and F. Dressler. "On the need for bidirectional coupling of road traffic microsimulation and network simulation." Proceedings of the 1st ACM SIGMOBILE workshop on Mobility models, 2008.

[9] C. C. Miller. "A beast in the field: The Google Maps mashup as GIS/2." Cartographica: The International Journal for Geographic Information and Geovisualization 41.3, pp. 187-199, 2006.

[10] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. B. Kopena. "Network simulations with the ns-3 simulator." SIGCOMM demonstration, 2008.

[11] "SUMO Wiki page", available online at: http://sumo.dlr.de/wiki/SUMO

[12] ns-3 Consortium. "ns3::RandomWaypointMobilityModel Class Reference" (2015), online at: http://www.nsnam.org/doxygen/classns3_1_1_random_waypoint_mobility_model.html

[13] T. Camp, J. Boleng, and V. Davies. "A survey of mobility models for ad hoc network research." Wireless communications and mobile computing 2.5, pp. 483-502, 2002.

[14] Developers' forum for Google Maps API, "Decoding polylines in Google Maps Directions API", available online at https://developers.google.com/maps/documentation/utilities/polylinealgorithm

[15] T. W. Leung. "Professional XML Development with Apache Tools: Xerces, Xalan, FOP, Cocoon, Axis, Xindice". John Wiley & Sons, 2004.

[16] C. Karney, "GeographicLib", online at http://geographiclib.sourceforge.net/

[17] D. Stenberg, "libcurl: The multiprotocol file transfer library", online at: http://curl.haxx.se/libcurl/

[18] J. Haerri, F. Filali, and C. Bonnet. "Performance comparison of AODV and OLSR in VANETs urban environments under realistic mobility patterns." Proceedings of the 5th IFIP mediterranean ad-hoc networking workshop, 2006.

[19] B. Raney, A. Voellmy, N. Cetin, M. Vrtic, and K. Nagel. "Towards a microscopic traffic simulation of all of Switzerland." The International Conference on Computational Science (ICCS 2002), Springer Berlin Heidelberg, pp. 371-380, 2002.