Simulating Smart Cities with DEUS

Marco Picone
Dept. of Information Engineering
University of Parma
Parma, Italy
picone@ce.unipr.it

Michele Amoretti
SITEIA.PARMA Laboratory
University of Parma
Parma, Italy
michele.amoretti@unipr.it

Francesco Zanichelli
Dept. of Information Engineering
University of Parma
Parma, Italy
francesco.zanichelli@unipr.it

ABSTRACT

Smart cities are envisioned to generate and consume overwhelming amount of data which can be harnessed to provide relevant information about their status so as to enhance the security and lifestyle of their citizens. Discrete event simulation is a powerful means to aid the design of the enabling ICT infrastructure for smart cities, in particular as a tool to predict the impact on user behaviors to the purpose of improving key urban business processes. Our general-purpose simulation environment, called DEUS, may well be a suitable candidate for coping with such type of analysis. In this paper we illustrate how DEUS has been used to simulate the effects of a peer-to-peer traffic information system, using a realistic mobility model applied to a medium/large city. The core engine of the simulation environment has been integrated with Google Map APIs, in order to allow real-time visualization of the simulated traffic with or without the traffic information system.

Categories and Subject Descriptors

I.6 [**Simulation and Modeling**]; C.2.4 [**Computer - Communication Networks**]: Distributed Systems; C.4 [**Performance of Systems**]

General Terms

Design,Performance,Verification

1. INTRODUCTION

According to a study of the United Nations [16], the fraction of the world's population living in cities is going to increase from current 50 to 70 percent, due to growth in the current urban population and migration from rural areas. Indeed, cities offer large economic, social, and political opportunities, as well as potential for significantly greater environmental sustainability. However, it is necessary to find new ways to manage complexity, to increase efficiency, to reduce expenses, and to improve quality of life. In other words, cities need to get smarter.

Progress lies in an accurate view across urban infrastructure, the right level of intelligence to optimize resources, and the ability to integrate information from all departments to anticipate and respond to events. Smarter city transformation relies on the use of powerful analytical techniques to extract insights from real-world events in order to improve urban business processes [12]. Creating and applying a unified information model gives the possibility to obtain a more complete picture of urban activity. The ability to understand how combinations of factors contribute to, *e.g.*, a rapid increase in the demand for water or an unusually high accident rate on a stretch of road in turn facilitates better operational decisions.

Multi-scale modeling & simulation (M&S) are necessary for the design and analysis of smart cities, that are particularly complex systems. The discrete event approach is particularly suited, allowing to focus on state changes in correspondence of meaningful events, thus reducing the execution time of simulations. To support this claim, in this paper we show how we can model a fundamental component of smart cities, namely the traffic information system, with our discrete event simulation environment, called DEUS [1]. A holistic simulation platform for emergent phenomena which include dependent systems does not exist yet. However, DEUS is flexible and general-purpose enough, to enable the effective simulation of several types of actors, devices, infrastructures, etc.

The paper is organized as follows. In section 2 we discuss related work on M&S of smart cities. In section 3 we recall the main features of DEUS, in particular those never presented before. In section 4 we present the package we have implemented to model mobile entities, such as cars, and the integration of DEUS core engine with Google Maps API. In section 5 we illustrate how such work has been used to simulate a smart vehicular network. Finally, in section 6 we conclude the paper with a discussion of presented work and an outline for future related research.

2. RELATED WORK

Being ”smart cities” a relatively recent concept, few simulation tools exist to support their design and analysis, and most of them are domain-specific.

Lugaric *et al.* [8] have proposed a simulation tool for analyzing emergent phenomena in smart grids, with the ability to simulate physical grid properties, communication infrastructure and information flow, and to model the behavior of human operators — to this purpose, an agent-based approach has been adopted. In the same domain, another tool based on software agents has been proposed by Karnouskos *et al.* [7], to simulate discrete heterogeneous devices that consume and/or produce energy, that are able to act autonomously and collaborate. The simulation included also 156 vehicles (52 at each city) and was running for more than 24 hours with all devices reporting their consumption every second to the database and the controller agent. The paper does not provide insights on the adopted mobility model and on the scalability of the proposed approach. The main interest of the authors appears to be in coupling simulations with real-world devices and connecting them to enterprise systems. To cope with scalability issues, discrete event simulation has been used by Molderink *et al.* [11]. Their tool allows to analyze the impact of different combinations of micro-generators, energy buffers, appliances and control algorithms on the energy efficiency, both within the house and on larger scale. More recently, we have used DEUS to study the integration of distributed energy generation and information systems [2].

Another domain that is gaining momentum, in relation with smart cities, is that of vehicular ad-hoc networks (VANETs). Deploying and testing VANETs involves high cost and intensive labor. Hence, simulation is a useful alternative prior to actual implementation. Simulations of VANETs often involve large and heterogeneous scenarios. Compared to MANETs, while simulating VANETs it is necessary to account for some specific characteristics found in a vehicular environment. For MANETs, the random waypoint model (RWP) is by far the most popular mobility model [18], but in a vehicular network, nodes (vehicles) can only move along streets, prompting the need for a road model. Network simulators that perform detailed packet-level simulation of source, destinations, data traffic transmission, reception, background load, route, links, and channels. Examples are ns-2/ns-3 [3], GloMoSim [9], SNS [17], JiST/SWANS [6], and GTNetS [5]. Non-commercial tools for traffic flow simulation have been evaluated and compared in a recent survey by Martinez *et al.* [10].

With respect the aforementioned tools, DEUS is more flexible, because of its general-purpose nature, allowing to consider almost any aspect of a complex system, like a smart city is. Such a property does not prevent to create reusable extensions of the DEUS APIs, as we show in the following sections.

3. OVERVIEW OF DEUS

Several free or commercial discrete event simulation (DES) tools are available, but no one appears to be sufficiently generic and flexible to support the analysis of complex systems at every scale. For this reason, we are developing a novel general-purpose, open-source DES environment, called *DEUS* [4, 1], characterized by extreme ease of use and flexibility.

DEUS is multi-platform, being developed with Java language. Its API allows developers to implement (by subclassing)

- nodes - the parts which interact in a complex system, leading to emergent behaviors: humans, pets, cells, robots, intelligent agents, etc.;
- events - *e.g.* node births/deaths, interactions among nodes, interactions with the environment, logs, etc.;
- processes - either stochastic or deterministic ones, constraining the timeliness of events.

A node may represent a dynamic system characterized by a set of possible states, whose transition functions may be implemented either in the source code of the events that can be associated to the node, or in the source code of the node itself. The core engine of DEUS places events in a queue that is implemented using a PriorityQueue, for which the insertion of a new event in the right position, according to the timestamp, costs $O(\log N)$, where N is the size of the queue. The timestamp of each event is computed by means of the process associated to the ”type” of the event. For example, events describing job arrivals to a server may be scheduled according to a Poisson process, for which the probability distribution of the waiting time until the next occurrence is an exponential distribution. DEUS provides a number of processes, and many others can be implemented by the user.

Multi-scale modeling of complex system can be achieved by defining nodes of different complexity and connecting them, *i.e.* making their instances refer to each other. To this purpose, the DEUS API provides a class called Peer that supports linking.

4. SIMULATION OF MOBILE NODES

Mobility models are massively used by mobile communication systems for predicting future user positions. They are crucial to test and evaluate for example vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I) or smart cities networking applications in real testbed environments. Those aspects are becoming highly relevant, considering the growing market of smartphones and mobile devices that are changing and reshaping the application design.

Starting from these motivations we decided to natively support mobility in our general purpose simulator extending the base node class and introducing the concept of Mobile Peer with a dedicated API that provides all necessary classes and methods to support mobility. This approach allows the developer to design and implement protocols and applications based on user mobility, starting from the classes offered by DEUS mobility package or extending them to be customized on specific requirements.

4.1 Mobility Model

Mobility Models (MM) describe the movement of mobile users, and how their location, velocity and acceleration change over time. Such Models are frequently used for several simulation purposes when new communication or navigation techniques are evaluated.

Designed mobility model follows the approach of Zhou et al. [19] where the key idea is to use switch stations (SSs) connected via virtual tracks to model the dynamics of vehicle and group mobility. Stations are connected to each other through virtual paths that have one lane for every direction, speed limitation associated with the street category and specific road density limit to model user/vehicle speed in jam conditions. When a new node joins the network, it first associates with a random SS, then it selects a new destination station and starts moving on the connection path between them. This procedure is repeated every time the user reaches a new SS and has to decide its next destination. Each switch station has an attraction/repulsion value that influences the user's choice for the next destination station. This value may be the same for each path in order to allow for random trip selection or could be configured by the developer according to the application field. Fig. 1 shows a schema of the SS Model and an example of simulative analysis that considers a square area around the city of Parma, with 20 switch stations inside and outside the city district.

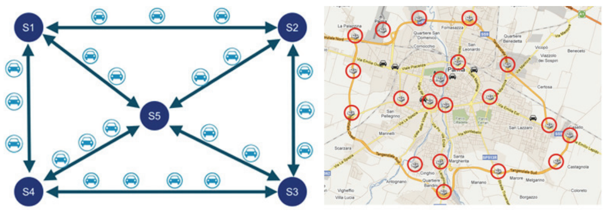

Figure 1: A schema of the SS Model (on the left), and an example of simulative analysis that considers a square area around the city of Parma, with 20 switch stations (on the right).

When SSs and virtual paths between them have been configured for the simulation each node extending MobileNode class starts moving on its selected path. Each movement is associated to a specific DEUS event (MovePeerEvent) that simulates the movement of a peer to the next location.

By defining $X_i(t)$ and $X_i(t+\Delta t)$ as node i's locations at time t and $t+\Delta t$, respectively, we can say that Δt is the time needed by i to move from the first to the second location and it is evaluated according to the distance between those points and to node speed. Given that, by knowing the node velocity at a certain instant, it is possible to put an event in the simulator queue that describes the node position changes along the path (Fig.2). When the event is picked up from the queue by the DEUS Engine, the location of the associated node is finally updated and a dedicated method is called to allow the developer to implement specific behaviors related to the position change of the node.

Within the presented approach, a general definition is needed for the speed of a Mobile Node, which may represent for example a pedestrian or a vehicle moving along a path. In order to do that, we rely on the concept of Speed Model, that computes the speed according to a set of parameters (the latter could be variable according to the model) describing the node characteristics and the state of the environment, such as the position of vehicles in the neighborhood, the actual speed, the maximum/minimum velocity, the maximum/minimum acceleration, the path length, and the node density on the track. DEUS provides an interface called ISpeedModel, that allows to define the Speed Model that the

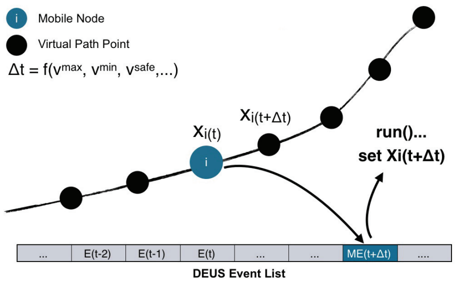

Figure 2: By knowing the node velocity at a certain instant, it is possible to insert an event in the simulator queue that describes the node position changes along the path.

user wants to use in its simulation. By default we provide the implementation of Fluid Traffic Model (FTM) [14]. It can be seen as an hybrid model, adopting a traffic system approach on a microscopic level. FTM describes the speed as a monotonically decreasing function of the vehicular density, forcing a lower bound on speed when the traffic congestion reaches a critical states.

4.2 MobDEUS Architecture

In section 4.1 we introduced the concept of mobility in DEUS and outlined the API provided to developers to implement their simulations involving mobile nodes. In order to offer a complete tool set, we have developed some external applications that are useful to configure and monitor the simulations.

Fig. 3 schematically illustrates the design of MobDEUS. Involved elements can be divided in two categories: the first one contains two external tools used to generate mobility scenario files needed by the DEUS Engine, and the second one is related to real-time monitoring of simulations.

The **Switch Station Web Editor (SSWE)** is a web-based tool written using Javascript and the Google Maps API, that allows to create and manage Switch Station configurations. The user can place on the map each SS according to its latitude and longitude, and generate, load and edit new or existing configuration files in order to be used with the simulator. DEUS provides a dedicated class called SwitchStationController to read, parse and load in a accessible list the configured SSs for the simulation.

The **Map Loader (ML)** is a Java application that communicates with Google Servers to retrieve paths between a list of Switch Stations. It takes as input file the SS list generated by the SSWE. Since the original received paths contain sparse geographic coordinates, each track is locally enriched by ML by computing coordinates between original points. The precision of this procedure can be defined by the user, by specifying the desired accuracy level (called *mobility precision (MP)*, expressed in meters) that he/she wants between two available geographic locations in the resulting file. The generated output contains the list of all virtual tracks and their coordinates and can be automatically read by DEUS Engine using the SwitchStationController class. In such a way the user is able to properly configure the simulation scenario and when the SwitchStationController instance has been created and has imported the user files, each MobileNode in the simulation uses such information for its mobility.

The **Real-Time Simulation Monitoring (RTSM)** is a web-based tool written using Javascript and Google Maps API, that allows to monitor active mobile nodes during a simulation. The DEUS mobile module provides a dedicated log event called LogNodeMapEvent, that periodically generates an XML file containing node positions and additional details about each node, like start/end Switch Station, neighbors information, actual speed, etc. Those data are placed on a map and each marker can be associated to a different icon, according to node type or status. By clicking on each marker, the developer obtains additional information about the actual status of the node, by visualizing the node profile that is written in the map file. Map data are automatically updated, thus allowing to monitor node behaviors during the simulation. Since DEUS is a completely open project, the Javascript code of the RTSM may be extended according to user preferences, for example to enrich visualized information (as shown in Fig. 4). The frequency of updates of our GMaps-based visualization tool may be lower than the frequency of simulated events (in this case we show only a subset of the events). In order to have full control over visualization, we plan to develop a standalone tool based on Open Street Map which will work offline using downloaded map portions.

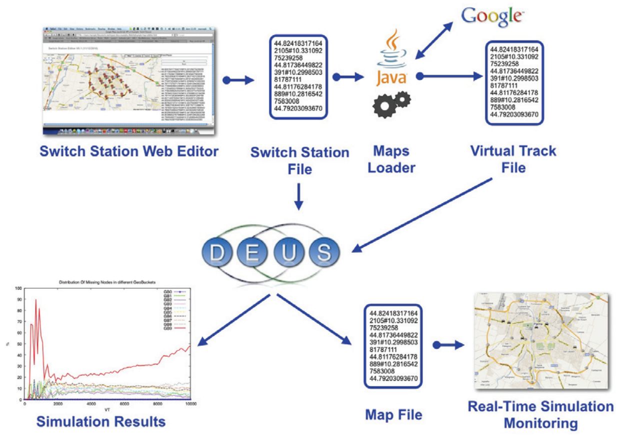

Figure 3: Simulation design of Traffic Information system for Smart Cities with MobDEUS.

5. EXPERIMENTAL WORK

We have used DEUS, and in particular the MobDEUS package, to design and evaluate a decentralized software architecture for smart cities, that includes different types of mobile/fixed nodes — data sources, storage nodes, data aggregators, user applications. Such nodes are organized in a structured peer-to-peer (P2P) overlay scheme called *Distributed Geographic Table (DGT)* [13], that takes into account their geographic position, thus enabling a number of location-based services. For example, we have simulated a DGT-based traffic information system that supports vehicular mobility, by suggesting alternative routes to avoid traffic jams or accidents. In this scenario, the knowledge of the in-

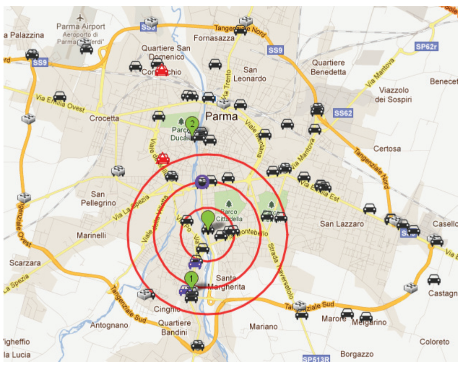

Figure 4: Real-Time Simulation Monitoring.

formation geographic position may be useful in computing the routes to be suggested to the users.

The DGT is a structured overlay scheme where each participant can efficiently retrieve node or resource information (data or services) located near any chosen geographic position. Every peer maintains a set of *geo-buckets* (GBs), each one being a (regularly updated) list of known peers sorted by their distance from the GP of the peer itself. GBs can be represented as K concentric circles, with increasing (application-specific) radii $\{R_i\}_{i=1}^K$ and thickness $\{r_i\}_{i=1}^K$, with $R_i = \sum_{j=1}^i r_j$. If there is a known node whose distance from the peer is larger than the radius of the outmost circle R_K, it is inserted in another list that contains the nodes outside the circle model. In such a system the responsibility for maintaining information about the position of active peers is distributed among nodes, for which a change in the set of participants causes a minimal amount of disruption.

Thanks to DEUS, we have been able to simulate the neighborhood discovery protocol, as well as data sharing strategies, considering both fixed and mobile peers. Regarding the latter, we have also simulated vertical handover, that refers to the possibility that mobile nodes switch connections among different access networks.

When measuring the accuracy of the protocol in building the neighborhood, performance metrics of interests are:

- *PMN*: Percentage of Missing Nodes in the neighborhood of a peer, with respect to those actually present in the area.
- *MR [msg/sec]*: Message Rate, *i.e.* the average number of messages received per second by each node.
- *NPE [Km]*: Node Position Error, *i.e.* the average distance between a peer's position reference in a geo-bucket and its actual position.

Several different peer systems have been simulated over a significant time span. The first one includes 1000 peers for a virtual time of 10 hours (corresponding to 10000 virtual time units) while the second one has doubled size (2000 peers) and time span (20 hours, that is 20000 virtual time units). In both cases the node set grows to full size during the first half of the simulation, after which only an insignificant number of peers enters and leaves the network.

The following table presents all considered cases.

Case	#Geo-buckets	GB thickness	#Nodes
1	10	1.5 Km	1000
2	5	3 Km	1000
3	10	0.5 Km	1000
4	10	0.5 Km	2000
5	10	1.5 Km	2000

Fig.5 shows the PMN for cases 1, 2 and 3, which refer to the first peer system (1000 nodes over 10 hours). We remark that the PMN remains moderate over all the simulated period, for all geo-bucket configurations. In particular for the first half of simulation, where many new nodes enter the system, the PMN value only grows up to around 5%, while it decreases significantly when the peer network reaches a more stable state.

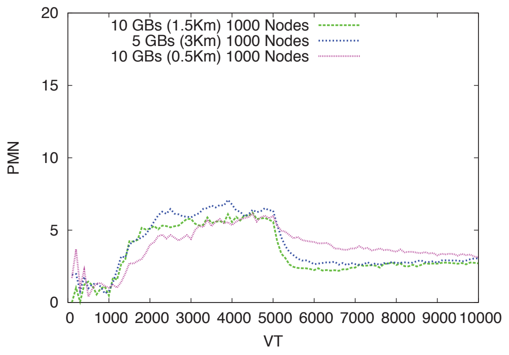

Figure 5: PMN - Evaluation of GB configurations (cases 1, 2 and 3) with 1000 nodes.

Fig.6 shows the evolution of MR, for cases 1, 2 and 3. Given that a larger covered area is potentially associated to a higher number of active peers, an increased number of known nodes must be contacted to obtain GP updates. For this reason, simulation results show that cases 1 and 2 have an increased MR value compared with case 3 where the covered area is smaller. In any case, the number of exchanged messages is very low, notwithstanding the fact this is a fully decentralized system where knowledge is maintained cooperatively by all available peers.

The same analysis has been carried out on the larger network (2000 active peers) network using two configuration of geo-buckets (cases 4 and 5) in order to assess the protocol behavior with a different distribution of nodes.

To avoid excessive bandwidth consumption, every peer communicates its position update to neighbors only if the displacement is higher than ϵ (Km). To assess the effects of ϵ variations on protocol performance, using a network of 2000 nodes, we have executed multiple simulations by varying ϵ between 0.1 Km to 1.35 Km in 0.25 Km steps. A low ϵ value means that updates of peer positions are performed very often when users change their locations (resulting in more events to be simulated), whereas a high value causes infrequent updates.

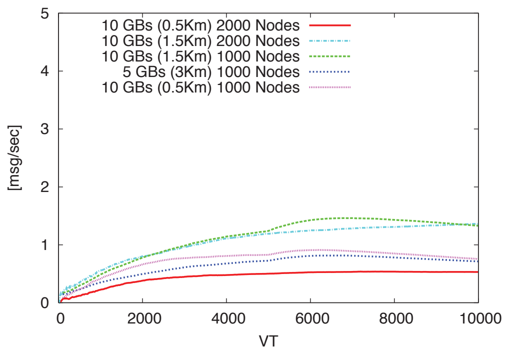

Figure 6: Global MR - Evaluation of GB configurations.

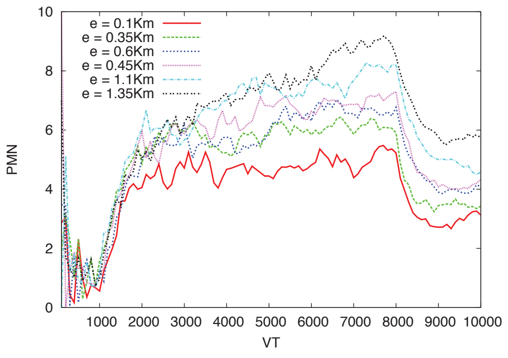

Figure 7: PMN - Evaluation with different values of the position update rate ϵ.

The accuracy of information owned by each peer is clearly related to the value of ϵ. Fig.7 shows the percentage of missing nodes with multiple ϵ values and we can see that there is a noticeable spread in the PMN results. This behavior is justified by the fact that a large ϵ value may lead to the erroneous exclusion or removal of a peer from the neighborhood, resulting into accuracy loss and inconsistency.

The analysis of the NPE (Fig.8) shows that the average error is slightly larger than the threshold as there is an additional little variation introduced by peers' mobility and information distribution among available nodes.

We are further validating the model and simulation results using a very recent implementation of the DGT protocol for the Android platform, running over PlanetLab. Preliminary results confirm simulated ones reported in this section.

6. CONCLUSIONS

In this paper we have shown how the design and analysis of smart cities can be effectively supported by M&S. We have recalled the main features of our DEUS simulation environment, and presented its new package for modeling mobile nodes and related events.

Regarding the performance of DEUS, figure 9 shows the real

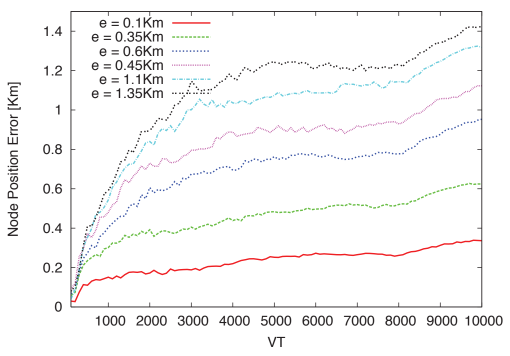

Figure 8: NPE - Evaluation with different values of the position update rate ϵ.

time that is necessary to simulate a smart city with 100 km of roads, considering different densities for the mobile peers carried by vehicles. With high densities, simulations take long time, for which it is necessary to run many of them in parallel on different machines. However, density values higher than 20 are unrealistic (they would mean that all cars are always in a traffic jam). The time of the simulation depends also on the mobility precision MP and on ϵ. Figure 9 refers to a high-precision configuration, with $MP = 10$ m and $\epsilon = 0.1$ km.

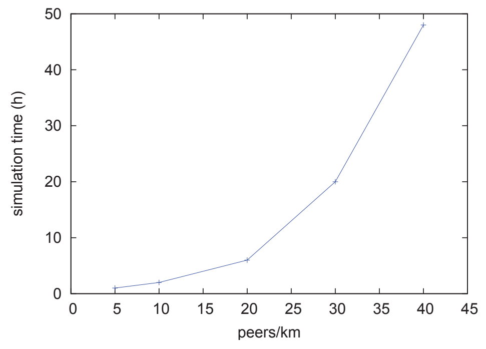

Figure 9: Simulation time for a smart city with 100 km of roads, considering different densities for the mobile peers and a high-precision configuration for the mobility model.

In the next future, we plan to simulate other mobility models and compare them with the one presented in this paper. We have developed a real implementation of our DGT overlay, that we are going to test in a real urban scenario. The performance of the real system will be compared with the results of the simulations.

7. REFERENCES

[1] M. Amoretti, M. Agosti, F. Zanichelli, *DEUS: a Discrete Event Universal Simulator*, Proc. of the 2nd ICST/ACM International Conference on Simulation Tools and Techniques (SIMUTools 2009), Roma, Italy, March 2009.

[2] M. Amoretti, *Towards a peer-to-peer hydrogen economy framework*, International Journal of Hydrogen Energy, Volume 36, Issue 11, June 2011, pp.6376–6386.

[3] G. Carneiro, H. Fontes, M. Ricardo, *Fast prototyping of network protocols through ns-3 simulation model reuse*, Simulation Modeling Practice and Theory, vol. 19, iss. 9, pp. 2063-2075, 2011.

[4] Distributed Systems Group, *DEUS project homepage*, http://code.google.com/p/deus/

[5] *The Georgia Tech Network Simulator (GTNetS)*, 2008. Available at: http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/

[6] *JiST/SWANS: Java in Simulation Time/Scalable Wireless Ad hoc Network Simulator*, 2004. Available at: http://jist.ece.cornell.edu/

[7] S. Karnouskos, T. N. de Holanda, *Simulation of a Smart Grid City with Software Agents*, European Modelling Symposium (EMS 2009), Athens, November 2009.

[8] L. Lugaric, S. Krajcar, Z. Simic, *Smart City - Platform for Emergent Phenomena Power System Testbed Simulator*, Innovative Smart Grid Technologies Conference Europe (ISGT Europe), Gothenburg, Denmark, 2010.

[9] J. Martin, GloMoSim. Global mobile information systems simulation library. UCLA Parallel Computing Laboratory, 2001. Available at: http://pcl.cs.ucla.edu/projects/glomosim/

[10] F. J. Martinez, C. K. Toh, J.-C. Cano, C. T. Calatafe, P. Manzoni, *A survey and comparative study of simulators for vehicular ad hoc networks (VANETs)*, Wirel. Commun. Mob. Comput., 2009.

[11] A. Molderink, M. G. C. Bosman, V. Bakker, J. L. Hurink, G. J. M. Smit, *Simulating The Effect on The Energy Efficiency of Smart Grid Technologies*, Winter Simulation Conference (WSCÕ09), Austin, TX, USA, December 2009.

[12] M. Naphade, G. Banavar, C. Harrison, J. Paraszak, *Smarter Cities and Their Innovation Challenges*, IEEE Computer, Vol.44 No.6, June 2011.

[13] M. Picone, M. Amoretti, F. Zanichelli, *Proactive Neighbor Localization Based on Distributed Geographic Table*, Proc. of the 8th ACM-SIGMM International Conference on Advances in Mobile Computing & Multimedia (MoMM 2010), Paris, France, November 2010.

[14] I. Seskar, S. Marie, J. Holtzman, J. Wasserman, *Rate of location area updates in cellular systems*, IEEE Vehicular Technology Conference (VTC'92), Denver, Colorado, USA, May 1992.

[15] United Nations, World Urbanization Prospects: The 2009 Revision - Highlights, 2010; http://esa.un.org/unpd/wup/doc_highlights.htm.

[16] K. Walsh, E.G. Sirer, *A staged network simulator (SNS)*, Computer Science Department, Cornell University, 2003. Available at: http://www.cs.cornell.edu/people/egs/sns/

[17] J. Yoon, M. Liu, B. Noble, *Random waypoint considered harmful*, IEEE INFOCOMM 2003, San Francisco, California, USA, April 2003.

[18] B. Zhou, K. Xu, M. Gerla. *Group and swarm mobility models for ad hoc network scenarios using virtual tracks*, IEEE Military Communications Conference (MILCOM), Monterey, California, USA, October 2004.