Simulating Mobile and Distributed Systems with DEUS and ns-3

Michele Amoretti[1], Marco Picone[2], Francesco Zanichelli[2], Gianluigi Ferrari[2]
(1) SITEIA.PARMA, (2) Information Engineering Dept.
Università degli Studi di Parma
43124 Parma, Italy
Email: {michele.amoretti, marco.picone, francesco.zanichelli, gianluigi.ferrari}@unipr.it

***Abstract*—Mobile and distributed systems are characterized by decentralized goals and control, with high levels of concurrency and asynchronous interaction. Their qualitative and quantitative analysis is usually based on discrete event modeling and simulation. As most simulation tools target a specific class of problems, only a few of them may be considered truly general-purpose, yet they can hardly support the analysis of distributed systems with thousands of nodes, characterized by a high level of churn (node joins and departures) and reconfiguration of connections among nodes. To fill this gap, a few years ago we started developing an open-source, general-purpose and discrete event simulation tool, called DEUS, which is application-level oriented, Java-based, and characterized by ease of use and flexibility. However, it does not provide any package for simulating networking layers and their implementation is not foreseen, since a number of specialized tools are already available. In this paper, we present a general methodology for achieving a more realistic DEUS-based simulation of mobile and distributed systems, by leveraging on ns-3, which is generally known as a highly reliable and complete open-source tool for the discrete event simulation of Internet systems. In particular, we describe our positive experience in using ns-3's LTE-EPC package to support the simulation of a peer-to-peer overlay scheme called Distributed Geographic Table (DGT), which allows mobile nodes to efficiently share information without centralized control.**

***Keywords*—Discrete Event Simulation, Mobile and Distributed Systems, DEUS, ns-3**

I. INTRODUCTION

Mobile and distributed systems are the result of the interconnection of several nodes — characterized by decentralized goals and control — that as a whole exhibit one or more properties (*i.e.*, behavior) which are not easily inferred from the properties of the individual parts. Such systems are complex, because the interactions of the nodes determine their future individual states and that of the system [1]. Moreover, they usually exhibit high levels of concurrency and asynchrony and their performance may be highly influenced by the changing environmental conditions of the environment (*e.g.*, if they move).

For the qualitative and quantitative analysis of such systems, discrete event modeling and simulation (in which time jumps from event to event) are usually adopted [2]. In order to choose the proper simulation environment, the following criteria should be taken into account: simulation architecture (the operation and the design of the simulator), usability (how easy the simulator is to learn and use), extensibility (the possibility to modify the standard behavior of the simulator in order to support specific protocols), configurability (how easily the simulator can be configured and with which level of detail), scalability (the ability to simulate how a P2P protocol scales with thousands, or more, nodes), statistics (how much the results are meaningful and easy to manipulate), reusability (the possibility to use the simulation code to write the real application).

By looking at the state of the art, it is evident that almost every simulation tool targets a specific class of problems. Only few of them may be considered general-purpose. Among these, the most advanced, in our opinion, is CD++ [3], which is a modeling environment that allows to define and execute Discrete Event System Specification (DEVS) models [2]. OMNeT++ is another well-known general purpose discrete event simulation tool, which has been publicly available since 1997 [4]. Like CD++, OMNeT++ is based on the concept of simple and compound modules. The user defines the structure of the model (the modules and their interconnection) using a topology description language called NED. OMNeT++ has been used in numerous domains from queueing network simulations to wireless and ad-hoc network simulations, from business process simulation to peer-to-peer network, optical switch and storage area network simulations.

Unfortunately, the above simulation tools are not particularly suitable for the analysis of distributed systems with thousands of nodes, characterized by a high level of churn (node joins and departures), and reconfiguration of connections among nodes. To fill this gap, in 2009 we started a project for the development of an open source, Java-based, general-purpose discrete event simulation tool, called DEUS [5]. To simulate a distributed system at the application level, DEUS is particularly convenient, because of its extreme ease of use and flexibility. However, it does not provide packages for simulating networking layers, and we do not foresee to implement them. For this reason, until now the scheduling of application-level events to simulate the exchange of messages among nodes has been necessarily configured by the user, using reasonable values — which can be considered as a naive approach.

In this paper, we present a general methodology for obtaining realistic DEUS-based simulation of mobile and distributed systems, leveraging on a highly reliable and complete open source tool for the discrete event simulation of Internet systems, namely ns-3 [6]. The latter relies on high-quality contributions of the community to develop new models, debug or maintain existing ones, and share results. In particular, we describe our positive experience in integrating ns-3's LENA LTE-EPC package [7] to support the network-aware simulation of a peer-to-peer overlay scheme called Distributed Geographic Table (DGT), which allows mobile nodes to efficiently share geo-referenced information without centralized control. To the best of our knowledge, OVNIS [8] is the only other tool which integrates ns-3 with a higher level discrete event platform, namely the SUMO road traffic simulator [9]. However, the only available release of OVNIS is the initial one, which includes an outdated version of ns-3.

The paper is organized as follows. Section II recalls the main features of DEUS and ns-3. Section III illustrates the methodology we propose for simulating mobile and distributed systems, using ns-3 to improve the realism of DEUS-based simulations. Section IV describes a challenging case study (regarding a peer-to-peer overlay network operating on top of LTE) we have addressed by means of the proposed methodology. Section V compares the results obtained with the proposed methodology with those obtained with a naive approach which models only the application layer. Finally, Section VI concludes the paper with a discussion of future work.

II. Overview of DEUS and ns-3

In this section, we summarize the main features of DEUS and ns-3, to pave the way for the presentation of our methodology for integrating them — which is illustrated in Section III.

A. DEUS in a nutshell

DEUS is a general-purpose discrete event simulation environment. It is a free, open source software project (with GPLv2 licensing). DEUS is multi-platform, being developed in Java. Its APIs allow developers to implement (by sub-classing) (i) *nodes*, i.e. the entities which interact in a complex system, leading to emergent behaviors such as humans, pets, cells, robots or intelligent agents; (ii) *events*, *e.g.*, node births and deaths, interactions among nodes, interactions with the environment, logs and so on; and (iii) *processes*, either stochastic or deterministic ones, constraining the timeliness of events.

Fig. 1 illustrates how DEUS simulation models, in terms of XML configuration files and Java code, are created (using also a Visual Editor), and then executed by means of the Automator and the Engine. The former allows to perform sensitivity analysis, by setting ranges for node and process parameters. The Engine is the core of DEUS, managing the event queue and the simulation loop.

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 associated to the node, or in the source code of the node itself. Multi-scale modeling of complex system can be achieved by defining nodes of different complexity and connecting them. DEUS comes with a library of predefined, common processes, and many others can be implemented by the user.

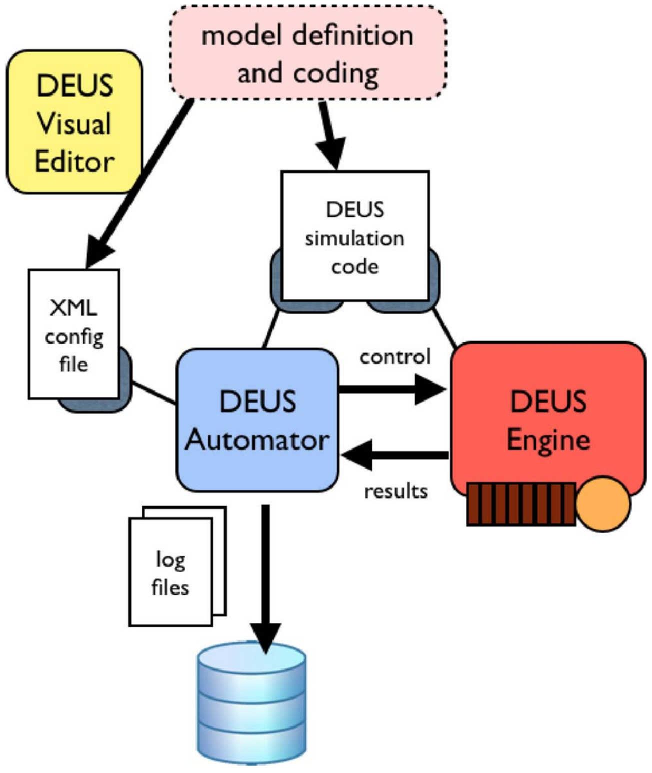

Figure 1: Discrete event simulation with DEUS.

Sample code is available on the web site of project DEUS.[a] Recently we have published an "USN resilience example", which simulates an Unstructured Supernode Network (USN) [10], *i.e.*, a peer-to-peer overlay network characterized by a group of peers, denoted as "supernodes," which have the responsibility of routing messages. Conversely, other peers ("leaf" nodes) are only resource providers and consumers, and need to connect to the supernode layer in order to publish and discover resources. In the considered scenario, if the lifetime L of a leaf node is longer than d, then at time d the leaf node becomes a supernode and connects to m other randomly selected supernodes. The simulation allows to evaluate the node degree distribution and the probability of isolations of the supernodes, as functions of m. On a Macbook Pro with 4 GB of 1067 MHz DDR3 RAM and a 2.4 GHz Intel Core 2 Duo processor, simulating a network with 1000 nodes takes a couple of minutes. With 10000 nodes it takes, on average, half an hour.

[a]http://code.google.com/p/deus/

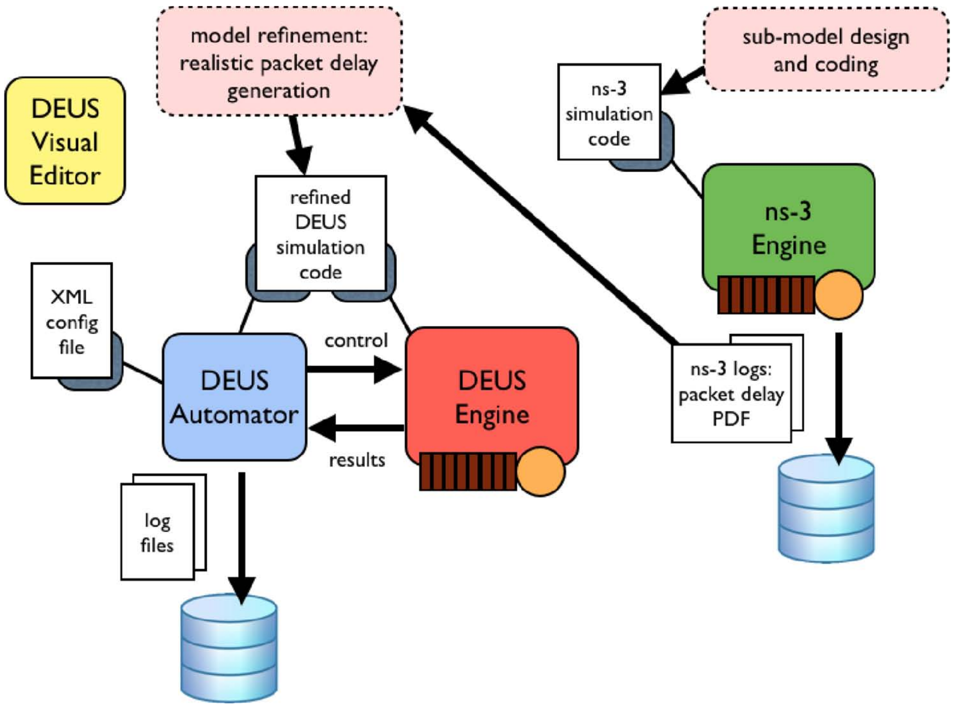
model refinement: realistic packet delay generation
sub-model design and coding
DEUS Visual Editor
ns-3 simulation code
refined DEUS simulation code
ns-3 Engine
XML config file
DEUS Automator
control
results
DEUS Engine
ns-3 logs: packet delay PDF
log files

Figure 2: Discrete event simulation with DEUS and ns-3.

Such an appreciable performance is due to the fact that only the application-level logic is being simulated, considering stochastic variables such as the inter-arrival time and the lifetime of the nodes, and neglecting communication delays. To simulate data exchange among peers, it would be necessary to define message delivery events with realistic timing processes. Of course it would be possible to implement, using the API of DEUS, the detailed simulation of the networking layers. Fortunately, this is not necessary, thanks to possibility to use dedicated simulation tools, such as ns-3, to fully characterize the communication delays.

B. *ns-3 in a nutshell*

ns-3 is a discrete-event network simulator for Internet systems. It is a free, open source software project (with GPLv2 licensing) organized around research community development and maintenance. Like its predecessor ns-2, ns-3 relies on C++ for the implementation of the simulation models. However, ns-3 no longer uses oTcl scripts to control the simulation, thus overcoming the problems which were introduced by the combination of C++ and oTcl in ns-2. Instead, network simulations in ns-3 can be implemented in pure C++, while parts of the simulation optionally can be realized using Python as well.

Moreover, ns-3 integrates architectural concepts and code from GTNetS [11], a simulator with good scalability characteristics. These design decisions were made at expense of compatibility — porting ns-2 models to ns-3 must be done in a manual way. Besides performance improvements, the simulator has an extended feature set. For example, ns-3 supports the integration of real implementations code by providing standard APIs, such as Berkeley sockets or POSIX threads, which are transparently mapped to the simulation.

Among the packages being developed for ns-3, the LENA LTE-EPC is particularly rich and efficient [7]. In the LTE-EPC simulation model, there are two main components:

- the LTE Model, which includes the LTE Radio Protocol stack (RRC, PDCP, RLC, MAC, PHY); these entities reside entirely within the User Equipment (UE) and the E-UTRAN Node B (eNB) nodes;
- the EPC Model, including core network interfaces, protocols and entities, which reside within the SGW, PGW and MME nodes, and partially within the eNB nodes.

III. Proposed Methodology

To simulate a distributed system with DEUS, it is necessary to write the classes which represent nodes, events and processes. Node may represent devices, servers, virtual machines, applications, etc. Events may be associated to specific nodes (*e.g.*, start, connection, disconnection, internally/externally triggered state change, stop, etc.), or involving several nodes (it is the case of logging events). To simulate a message delivery from one node to another, it is necessary to define the sender, the destination and to schedule a ”delivered message” event in the future (in terms of virtual time of the simulation). The scheduling time of such an event must be set using a suitable process, selected among those that are provided by the DEUS API, or defined by the user, possibly.

For example, if the purpose of the simulation is to measure the average delay of propagating multi-hop messages within a network of nodes (*e.g.* a peer-to-peer network), the value of each link's delay must be realistic, taking into account the underlying networking infrastructure. In particular, if the communication is wireless, estimating the delay of point-to-point communication is a challenging task.

The direct integration of DEUS with ns-3, with the former that "calls" the latter to compute a delay value every time a node must send a message to another node, taking into account current surrounding conditions, is unpractical and would highly increase the simulation time. Instead, a more effective and efficient solution (illustrated in Fig. 2) includes the following steps:

1) identify the main sub-system types, each one being characterized by specific networking features;
2) with ns-3: create detailed simulation models of the sub-systems (*i.e.*, sub-models), and measure their characteristic transmission delays;
3) with DEUS: simulate the whole distributed system, with refined scheduling of communication events, taking into account the transmission delays computed at step 2.

For example, if the overlay network relies on a cellular network, the sub-model to be characterized with ns-3 could be a set of cells. Its size should be significantly large, with respect to the system to be simulated with DEUS. If such a system is a peer-to-peer network, the end-to-end communication among couples of peers could span few or many cells, depending on the overlay scheme. Multi-cell communication may have a very high data rate, in case base stations are connected by optical fibers [12]. However, inter-cell interference and horizontal handover could be taken into account, when simulating mobile nodes. Moreover, the simulation of each cell should take into account the presence of other mobile nodes, that are not directly involved in the distributed application of interest, but consume significant resources. Finally, the same sub-system could be simulated with different geographic conditions, *e.g.* in a city (with small cells, buildings, and noisy channel), or in a rural area (with larger cells and a less disturbed channel).

For the case study illustrated in next section, we have modified the C++ class which creates the logs for the RLC protocol, in the LENA LTE-EPC package. The modified class logs a discretized probability density function (PDF) of the RLC packet delay. The latter is then used to generate realistic packet delays in the DEUS-based simulations, using the well-known *inversion method* [13], which is based on the inverse probability theorem. The main steps are:

- choose the cumulative distribution function $F(x)$ of the random variable to be sampled;
- generate a set of uniform random numbers such that $R \sim U(0,1)$;
- compute the random variate $X_i = F^{-1}(R_i)$.

To this purpose, the discretized PDF of the RLC packet delay is approximated either with a Multimodal Gaussian PDF (whose inversion has a high computational cost, unfortunately), or with a histogram PDF (whose inversion is straightforward).

IV. Case Study

We applied the proposed methodology to the modeling and simulation of the Distributed Geographic Table (DGT), which is a peer-to-peer overlay scheme with the main objective to provide support for mobile node localization. Compared to centralized localization approaches, the DGT is more scalable, since its performance (in terms of responsiveness, completeness and robustness) remains valuable also for a large number of nodes, and when the nodes' dynamics are very high [14]. In a DGT-based 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.

Every peer maintains a set of geo-buckets (GB), each one being a (regularly updated) list of known peers sorted by their distance from the Global Position of the peer itself. GBs can be represented as concentric circles, each one having a different (application-specific) radius and thickness. The distance between two DGT peers is defined as the actual geographic distance between their locations in the world. The neighborhood of a geographic location is the group of nodes located inside a given region surrounding that location.

The main service provided by the DGT overlay is to route requests to find available peers in a specific area, *i.e.*, to determine the neighborhood of a generic global position (Fig. 3). The routing process is based on the evaluation of the region of interest centered in the target position. The idea is that each peer involved in the routing process selects, among its known neighbors, those that presumably know a large number of peers located inside or close to the chosen area centered in the target point. If a contacted node cannot find a match for the request, it does return a list of closest nodes, taken from its routing table. This procedure can be used both to maintain the peer's local neighborhood and to find available nodes close to a generic target.

Further details about the DGT can be found, for example, in recent articles by Picone *et al.* [14], [15]. Simulation results presented there were obtained by means of a DEUS simulation model, integrated with Google Maps for having a realistic characterization of the urban environment (the city of Parma). However, simplistic assumptions on the packet transmission delay were made.

To better characterize the communication among DGT peers in the urban environment, we defined the sub-model illustrated in Fig. 4, using ns-3 with the Lena LTE-EPC package.[b]

[b]We used the latest version, released the 23rd of January 2013.

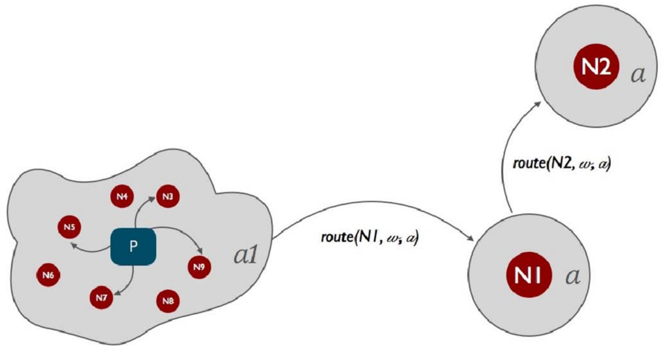

Figure 3: Propagation of a query between nodes to retrieve the neighborhood of a local or remote region of interest.

The latter provides (1) the E-UTRA part of the Long Term Evolution (LTE) technology, dealing with PHY, MAC and Scheduler functionalities, and (2) support for the LTE RLC and PDCP protocol, together with EPC data plane features, such as the S1-U interface and the SGW and PGW entities. Shortly, such an ns-3 package supports the detailed simulation of end-to-end IP connectivity over LTE-EPC.

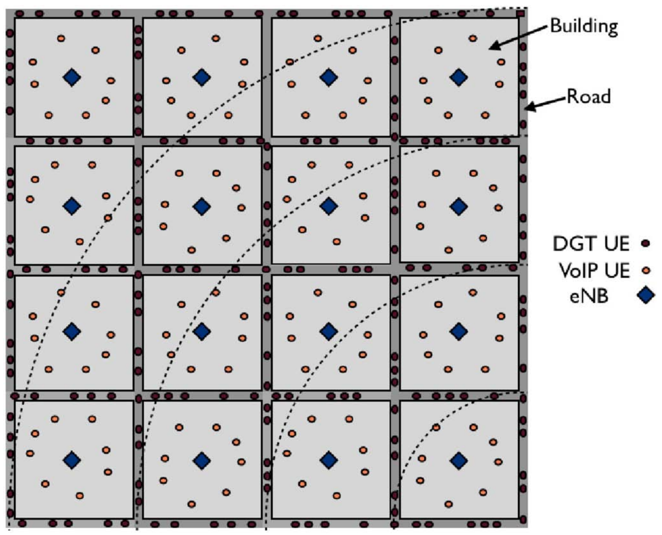

Figure 4: Bird's-eye view of the simulated scenario, with $n = 200$ DGT nodes and $v = 96$ other UEs randomly placed within the buildings. The geo-buckets of the DGT node in the bottom right corner of the map are also drawn, to show that the side length of the considered area equals the GB radius.

We considered DGT peers having GBs with radius of 2 km. Thus, we defined a square area having side length $l = 2$ km, with a grid of $r = 10$ roads (5 in N-S direction, and 5 in W-E direction) and vehicles running over them (with linear density $\delta = 10$ vehicles/km). Drivers are provided with User Equipments (UEs), which execute a DGT-based application. The total amount of DGT UEs is $n = r\delta l = 200$.

Parallel roads are spaced by $l/4 = 0.5$ km. Between each pair of parallel roads, there are four large buildings with squared area, each one having seven floors. Table 1 describes such buildings in detail. Randomly located within each building, there are $v/16$ other UEs, where v is their total amount. The pathloss model is ns3::BuildingsPropagationLossModel.

Table 1: Buildings

Building #	Parameters	Values
1..16	$x_{\min}$ [m]	100
	$x_{\max}$ [m]	300
	$y_{\min}$ [m]	100
	$y_{\max}$ [m]	300
	$z_{\min}$ [m]	0
	$z_{\max}$ [m]	21
	# floors	7
	# walls type	ConcreteWithWindows

Table 2: eNBs and UEs

Device type #	Parameters	Values
eNB	UlBandwidth [RB]	50
	DlBandwidth [RB]	50
	DlEarfcn	50
	UlEarfcn	18100
	z [m]	23
	Tx Power	49
	Noise Figure	5
UE	Tx Power	23
	Noise Figure	9

On top of each building, exactly in the middle, there is an eNB, *i.e.* a base station which serves a subset of the $n+v$ UEs. Table 2 reports the configuration parameters for the eNBs and the UEs. Regarding the eNBs, they have FDD paired spectrum, with 50 Resource Blocks (RBs) for the uplink, which means a nominal transmission rate of 50 Mbps, and the same for the downlink — like currently deployed LTE systems.

DGT UEs use UDP to send four types of DGT packets to each other. The first type, called Descriptor, is for neighborhood consistency maintenance purposes. Such a packet has the following structure:

- key: int (4 bytes)
- timestamp: float (4 bytes)
- lat: double (8 bytes)
- lng: double (8 bytes)

where key is the identifier of the peer in the DGT overlay network, timestamp is the current time, and lat/lng indicate the location of the node. Considering also the 12 bytes header, the size of the DGT packet is 36 bytes.

The second type of packet is the Lookup Request, which is used to search for remote nodes placed around a specified location. Its structure is the following:

- senderKey: int (4 bytes)
- lat: double (8 bytes)
- lng: double (8 bytes)

where senderKey is the identifier of the peer in the DGT overlay network that sends the request, and lat/lng indicate the location of interest. With the 12 bytes header, the size of such a DGT packet is 32 bytes.

The third packet type is the Lookup Response, which is sent by a DGT node as a reply to a lookup request, if the node owns the searched resource / information. The structure of the packet is the following:

- senderKey: int (4 bytes)
- lat: double (8 bytes)
- lng: double (8 bytes)
- descriptors: Descriptor[20] ($\leq$ 480 bytes)

where senderKey is the identifier of the peer in the DGT overlay network that sends the response, lat/lng indicate its location, and descriptors is a list of maximum 20 node descriptors. Considering the 12 bytes header, the maximum size of such a DGT packet is 512 bytes.

Finally, traffic information packets have the following structure:

- trafficMessage: String (30 bytes)
- senderDescr: Descriptor (24 bytes)
- ttl: float (4 bytes)
- range: double (8 bytes)

where trafficMessage is the message to be transmitted (*e.g.*, "traffic jam"), senderDescr is the descriptor of the sender DGT node, ttl is the time to live of the message, *i.e.*, the number of re-propagations it can be subject to, and range indicates the radius of the dissemination circle, which spatially limits the forwarding process.

We set an inter-packet interval of 50 ms for all types of DGT messages. Thus, the maximum rate is $512 \times 20 \simeq 10$ kB/s, while the minimum is $32 \times 20 = 0.64$ kB/s. In a dynamic DGT (the one we simulate with DEUS), packets are not sent periodically. For example, descriptors are sent only every ϵ meters. Lookup requests are sent only when necessary, as well as lookup responses. Traffic information messages are sent only when something interesting can be communicated to the other nodes (for example, a traffic jam or an incident).

The other UEs transmit and receive VoIP packets (using UDP) with a remote host located in the Internet. Such packets have a 12 bytes header and a 13 bytes payload, and inter-packet interval of 20 ms (we considered the AMR 4.75 kbps codec).

Each eNB has a scheduler which allocates RBs (which are the smallest elements of resource allocation) to users for predetermined amount of time. In these simulations, the Proportional Fair scheduler is used (ns3::PfFfMacScheduler), which tries to maintain a balance between two competing interests: trying to maximize total wireless network throughput while at the same time allowing all users at least a minimal level of service.

V. RESULTS

The ns-3 simulations were executed on a Ubuntu Linux 11.10 x86_64 machine with 16 GB of RAM and double quad core processor Intel(R) Xeon(R) Intel Xeon E5504 2.00 GHz. Each simulation was repeated with 20 different seeds for the random number generator.

For the DGT packet flow, we analyzed the uplink and downlink delays — to this purpose, we modified the logger of the LTE LENA package in ns-3 (as previously stated, in section III). The probability density function (PDF) of the uplink delay is basically a delta function, centered on 4 ms (Fig. 5). Instead, the PDF of the downlink delay can be approximated by a multimodal function, with three peaks (Fig. 6), with the following formula:

$$f_d(x) = 0.05g(x,9,2) + 0.62g(x,116,32) + 0.33g(x,172,28)$$

where $g(x,\mu,\sigma)$ is the normal PDF with mean μ and standard deviation σ (in milliseconds). The observed packet loss was less than 2%.

We further investigated the statistics of the downlink delay, by performing the following tests. We configured one DGT node (referred as probe node, from now on) in order to send only small packets (32 bytes), while the other send all possible packets (as described in section IV. We ran the simulations and we plotted the PDF of the delay observed by the probe node. We repeated the experiment by configuring the probe node in order to send only large packets (512 bytes). We obtained the same PDF of the previous test. Such a PDF matches the multimodal one observed in the first experiment (the one which is approximated by $f_d(x)$). Thus, the delay distribution is not affected by the size of the packet, if all nodes send packets whose type is randomly selected.

We performed the probe test also in other two cases, namely when all nodes always send small packets, and when they all send large packets. In the first case, the PDF is approximated by a normal density function with $\mu = 9$ ms and $\sigma = 2$ ms (the first peak of $f_d(x)$). In the second case, the resulting PDF is approximated by a bimodal PDF, whose peaks correspond to the second and third peaks of $f_d(x)$.

To include such an important result into the DEUS simulation model of the DGT, we have refined the latter by means of a new algorithm for sending messages between DGT nodes:

```
if (msgType = "descriptor") then
    msgSize ← 36
else if ( thenmgType = "lookup request")
    msgSize ← 32
else if ( thenmsgType = "lookup response")
    msgSize ← 512
else if ( thenmsgType = "traffic information")
    msgSize ← 76
end if
msgDelay ← Multimodal($\mu_1$,$\mu_2$,$\mu_3$,$\sigma_1$,$\sigma_2$,$\sigma_3$,$w_1$,$w_2$,$w_3$)+$D_{up}$
sendMessage(msgType, msgSize, msgDelay)
```

where $\mu_1 = 9$ ms, $\mu_2 = 116$ ms, $\mu_3 = 172$ ms, $\sigma_1 = 2$ ms,

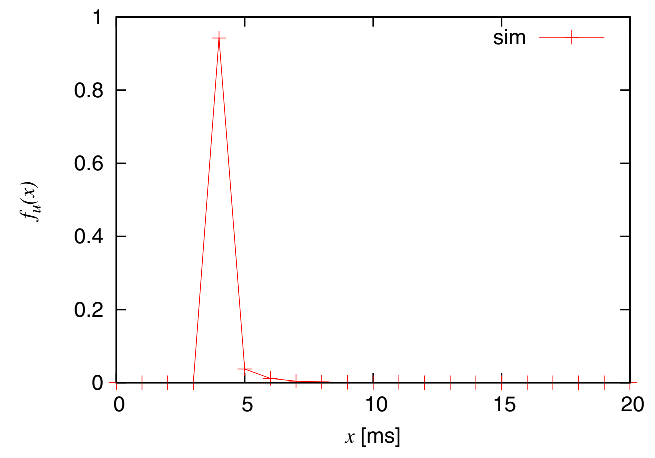

Figure 5: PDF of the uplink delay for DGT packets, obtained from ns-3.

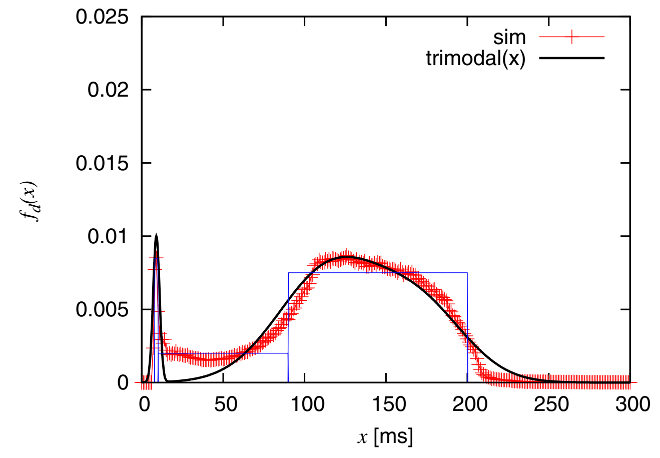

Figure 6: PDF of the downlink delay for DGT packets, obtained from ns-3.

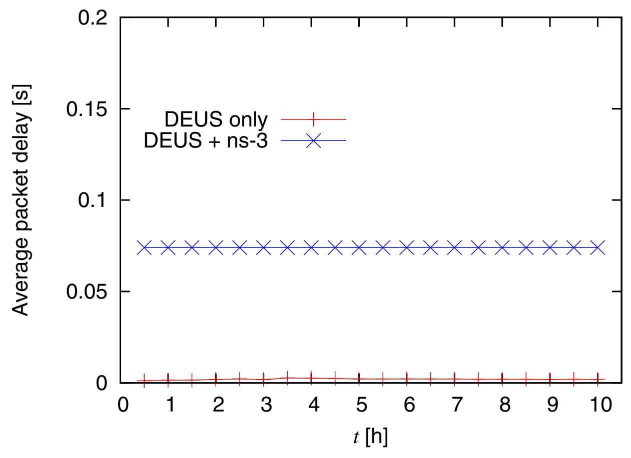

Figure 7: Average packet delay, measured with DEUS, for the simulated DGT overlay network with 1000 vehicles.

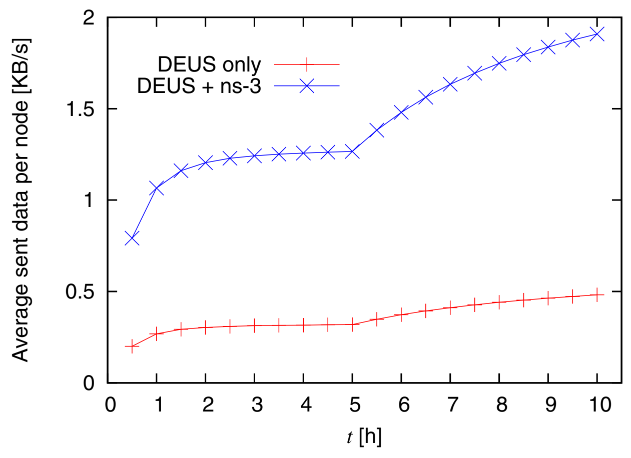

Figure 8: Average amount of sent data per node, measured with DEUS, for the simulated DGT overlay network with 1000 vehicles.

$\sigma_2 = 32$ ms, $\sigma_3 = 28$ ms, $w_1 = 0.05$, $w_2 = 0.62$, $w_3 = 0.33$ and $D_{up} = 4$ ms. The Multimodal() function is our porting of the Univariate Multimodal Generator of random Numbers (UMGRN) implemented by A. Suinesiaputra for MATLAB [16].

The proposed solution is a considerable improvement with respect to our previous DEUS-based DGT simulation model, which used, for every transmission, an exponential delay with mean value obtained by considering the nominal uplink and downlink.

We simulated a DGT overlay with 1000 mobile vehicles, over a period of 10 hours. In the first half of such a period, the network grows from 0 to 1000 nodes. In the second half, the size of the network remains stable. We logged the average packet delay and amount of sent data per node, computed on the whole overlay network. Fig. 7 and 8 compare the results obtained with the old simulation model, and those obtained with the refined one.

As we expected, in the refined model the average delay is higher than the one obtained with the naive model, which is based on nominal uplink and downlink values. Also the average amount of sent data is higher, because in the refined model we take into account also the header of the packets (12 bytes are 1/3 of Descriptor packets, which are the most frequently sent).

The DEUS simulations were executed on a Macbook Pro with 4 GB of 1067 MHz DDR3 RAM and a 2.4 GHz Intel Core 2 Duo processor. Unfortunately, generating delays by means of the Multimodal() function is time-consuming (each simulation run may take up to 15 hours). An alternative approach is to use the histogram approximation illustrated in Fig.6. With the latter, we obtained almost the same results given by the simulations based on Multimodal(), but in $1/10$ of the time — the same time required by the old simulation model. Thus, in future research works we will definitely use the histogram approximation.

VI. Conclusion

In this paper we have described a general methodology for obtaining realistic simulations of mobile and distributed systems, leveraging on DEUS and ns-3. The former allows to easily model and simulate application-level mechanisms and protocols, involving a large number of nodes with complex dynamics, while the latter is one of the best tools for simulating Internet systems down to the physical layer. We have illustrated a case study regarding a peer-to-peer overlay scheme, called DGT, whose main objective is to provide support for mobile node localization. To improve the characterization of the communication among DGT nodes, we have modeled and simulated a representative sub-system with ns-3, using the LTE-EPC package. Obtained packet delays have allowed to refine the DEUS-based simulation with 1000 mobile nodes.

Regarding future work, we plan to realize even more detailed simulation models using ns-3, not only for simulating the DGT but also other types of mobile and distributed systems. In particular, we are interested in applying the proposed approach to the study of Mobile Clouds, *i.e.* systems in which mobile applications dynamically offload their tasks, to preserve the battery charge of their devices, or simply to exploit the high performance of the Cloud.

References

[1] C. Gershenson and F. Heylighen, “How can we think the complex?” in *Managing Organizational Complexity: Philosophy, Theory and Application*, K. Richardson, Ed. Charlotte, NC, USA: Information Age Publishing, 2005, pp. 47–71.

[2] B. P. Zeigler, H. Praehofer, and T. G. Kim, *Theory of Modeling and Simulation, 2nd Ed.* Academic Press, 2000.

[3] G. Wainer, “CD++: a toolkit to develop devs models,” *Software - Practice and Experience*, vol. 32, no. 13, pp. 1–46, Nov. 2002.

[4] A. Varga and R. Hornig, “An overview of the OMNeT++ simulation environment,” in *1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (SIMUTools 2008)*, Marseille, France, Mar. 2008.

[5] M. Amoretti, M. Agosti, and F. Zanichelli, “DEUS: a discrete event universal simulator,” in *2nd ICST/ACM International Conference on Simulation Tools and Techniques (SIMUTools 2009)*, Rome, Italy, Mar. 2009.

[6] “ns-3 official homepage,” http://www.nsnam.org, 2013.

[7] N. Baldo, M. Requena-Esteso, J. Nin-Guerreo, and M. Miozzo, “A new model for the simulation of the LTE-EPC data plane,” in *5th ICST/ACM International Conference on Simulation Tools and Techniques (SIMUTools 2009)*, Sirmione, Italy, Mar. 2012.

[8] Y. Pigne, “The OVNIS platform,” http://ovnis.gforge.uni.lu, 2010.

[9] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, “SUMO - Simulation of Urban MObility: An overview,” in *SIMUL 2011, The Third International Conference on Advances in System Simulation*, Barcelona, Spain, October 2011, pp. 63–68.

[10] M. Amoretti, “A modeling framework for unstructured supernode networks,” *IEEE Commun. Lett.*, vol. 16, no. 10, pp. 1707–1710, Oct. 2012.

[11] G. Riley, “Large scale network simulations with GTNetS,” in *Winter Simulation Conference)*, New Orleans, Louisiana, USA, Dec. 2003.

[12] A. Nagate, K. Hoshino, M. Mikami, and T. Fujii, “A field trial of multicell cooperative transmission over LTE system,” in *IEEE International Conference on Communications (ICC 2011)*, Kyoto, Japan, Mar. 2011.

[13] M. Guizani, A. Rayes, B. Khan, and A. A-Fuqaha, *Network Modeling and Simulation, 1st Ed.* Wiley, 2010.

[14] M. Picone, M. Amoretti, and F. Zanichelli, “Evaluating the robustness of the DGT approach for smartphone-based vehicular networks,” in *Proceedings of the 2011 IEEE 36th Conference on Local Computer Networks*, ser. LCN ’11, 2011, pp. 820–826.

[15] ——, “Proactive neighbor localization based on distributed geographic table,” in *Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia*, ser. MoMM ’10, 2010, pp. 305–312.

[16] A. Suinesiaputra, “Univariate Multimodal Random Number Generator for MATLAB,” http://www.mathworks.com/matlabcentral/fileexchange/6488-univariate-multimodal-random-number-generator/content/umgrn.m, Dec. 2004.