Design and Quantitative Assessment of a Novel Hybrid Cloud Architecture for VANET Simulations

Hector Agustin Cozzetti, Giuseppe Caragnano, Klodiana Goga, Daniele Brevi, Olivier Terzo, Riccardo Scopigno
Istituto Superiore Mario Boella
10129 - via P.C. Boggio 61 - Turin, Italy
Email: cozzetti,caragnano,goga,brevi,terzo,scopigno@ismb.it

***Abstract*—Vehicular Ad-hoc NETworks (VANETs) are ad hoc networks aimed at improving the safety and efficiency of transportation in the near future. Despite the availability of results from field-trials, simulations still play an unequalled role in the comprehensive understanding of complex and crowded VANET scenarios. Even more, VANET simulations are typically computationally intensive problems and lend themselves for execution on distributed systems. This paper presents a new architecture optimizing the scheduling and execution of a batch of simulations over a hybrid cloud. Results reveal that, in case of multiple simulations to be executed, the overall performance can deeply benefit from a distributed approach, reducing time and costs.**

I. Introduction

The Intelligent Transportation Systems (ITS) technology is meant to improve the traveling experience, for instance by increasing the safety of transportation and the effectiveness of traffic management, and enhancing the environmental impact. For all these goals, Vehicular Ad-Hoc Networks (VANETs), that is wireless ad-hoc networks among vehicles, are expected to play a key role in the close future: the exchange of information between cars will widen the horizon and prevent accidents, jams and pollution, leveraging mutual coordination.

Currently, the standardization of VANETs has reached a nice stage but, despite this, only partial experimental data are available. In fact, on-field experiments are often too expensive and can not be carried out in fully crowded scenarios.

Nonetheless, due to complex phenomena involved in VANETs (high mobility, large number of vehicles and harsh environmental conditions), protocols and applications need to be extensively tested in order to guarantee reliable solutions. For this reason, network simulators still play a vital role for VANETs. In fact, simulations can support both the protocol design and the subsequent evaluation phases, providing results and feedback, under a wide set of conditions, at a lower cost than experiments. Many tools exist for this purpose, including NS-2 and NS-3, QualNet, OMNET++ [1]; all of them are more or less prone to scalability issues [2]. In principle, a distributed approach could optimize the computation time both for a single and for a set of simulations.

The objective of this work is to fill this gap and propose a new architecture based on a virtualised cloud-computing environment for the optimal scheduling of a batch of simulations over a hybrid cloud environment. This solution will be demonstrated to be feasible and to improve the performance achieved by the currently available methodologies. All these features help the final users reduce time and costs of simulations.

The remainder of the paper is organized as follows. The state of the art of network simulations and their characteristics are covered in sect. II. In sect. III the proposed architecture based on hybrid cloud computing is introduced; the exhaustive evaluation of the proposed solution is shown in sect. IV, while conclusions and future works are finally discussed in sect. V.

II. Simulation of VANETs: Characteristics and Issues

The incumbent standard for VANETs is based on a custom adaptation of WLAN, known as IEEE 802.11p [3]. With the exception of some limited regional differences in the regulation of channels, emitted power and multichannel environment, VANETs will mostly be worldwide solutions. This explains the global extent of the field trials (a.k.a. *field operational tests*) whose results typically cover multiple environmental settings and are set in worldwide locations [5]. The rationale for trials is to validate VANET solutions in reality, since most simulation models subtend some initial assumptions which may be simplistic (*e.g.* neglected Doppler effect, simplified propagation models) and lead to arguable results.

So one might conclude that simulations are not worthy anymore. However, there are at least two reasons which strongly motivate the continuous recourse to it and, consequently, the never-ending improvement of simulation tools. First of all, scalability has been recognized as a potential criticality of VANETs; on the other hand, given the limited availability of transceivers, it would be impossible to extensively study this issue with an experimental approach. Even more, the simulations have one potential which measurements cannot equal: each event can be decomposed into causes and events can be mutually correlated, eventually gaining the possibility to interpret each phenomenon revealed by the results.

Despite the recent efforts aimed at the optimization of simulations, VANET simulations are characterized by a heavy computational load. The load can be ascribed to the investigation of scalability (itself involving a large number of vehicles) and to the large number of phenomena subtended by simulations (*e.g.* mobility patterns, propagation phenomena, protocols mechanisms). Even more, some recent tools made the picture more complex adding the possibility to have a joint mobility and network simulation: with iTetris [6] NS-3 and SUMO are put together, so that mobility is not given

978-1-4673-1881-5/12/$31.00 ©2012 IEEE

as an input to the network simulator, but rather jointly run, with mobility patterns being influenced by traffic management messages exchanged between vehicles. While the realism of simulation is further improved, the load gets increasingly high.

This context itself substantiates the search for new tools addressing the optimization of the computational load involved by VANET simulations. Grid and cloud techniques could be powerful resources. At the time of this writing only the case of Omnet++ is known [7]; however it poorly supports VANET and wireless, which set the most challenging scalability simulation problems and constitute the intended goal of this study.

Distributed resources can be exploited also for another purpose. Frequently, VANETs require comparison of several scenarios *(i)* to deduce a preferred setting (*e.g.* transmitted power depending on the congestion state of the VANET), *(ii)* to identify a critical behavior (*e.g.* when congestion occurs a given setting) or *(iii)* to compare the effects of distinct propagation models or environmental settings (*e.g.* urban *vs.* highway). In all these cases, a large number of simulations must be run. As a consequence, distributed (cloud) resources can be vital for reducing the simulation time by spreading distinct simulations on the available platforms; in fact, as demonstrated in sect. IV, only a proper *smart* scheduling of the simulation tasks can optimize the computation, increasing the overall performance. Consequently, an architecture for the scheduling of simulations over a hybrid cloud is here proposed.

III. Proposed Hybrid-Cloud Scheduling Architecture for Simulations

In order to perform a large set of VANET simulations, a considerable amount of computing power and time is required. In this paper, a novel hybrid cloud architecture [8] is proposed, specifically to improve the performance of simulations, based on a virtualised cloud computing environment. The model implemented in this study is optimized for NS-2 [9], a network simulator widely known in the scientific community; however, for the sake of clarity, the proposed solution can be easily customized to support any other simulation software: this characteristic can certainly help the diffusion of this new approach and tool.

Cloud computing hides the complexity of IT infrastructure management from its users. At the same time, cloud computing platforms provide high performance, massive scalability and reliability. This system (fig. 1) has been developed according to the model of hybrid cloud computing [8], enabling the execution of large number of simulations in parallel. A hybrid architecture is defined as a mix of two possible resources: (*i*) *Virtual Machines* (VMs) belonging to a *private cloud*-platform; (*ii*) VMs available from a *public cloud*-platform (here Amazon EC2 was selected as public solution).

A. Amazon Elastic Compute Cloud (EC2) - Public Cloud

The Amazon Elastic Compute Cloud is an IaaS (Infrastructure as a Service)[8] cloud computing service, part of Amazon's cloud computing platform called Amazon Web Services (AWS). In EC2 users can boot an Amazon Machine Image (AMI) through a web service and create their own virtual machine (called "instance"). Amazon EC2 is based on the XEN virtualization technology (*para-virtualization*) and sizes instances based on EC2 Compute Unit [12] (ECU)[1]. An EC2 instance can be launched selecting between two types of storage for its root device:

- *Instance store* (ephemeral storage)[10] - persists only during the life of the instance.
- *EBS* (Elastic Block Storage)[11] - data on the root device persist independently of the lifetime of the instance. The user can stop and restart the instance at a successive time.

All in all, Amazon EC2 allows users to rent virtual machines (instances): the cost depends on several parameters like the type of instance and the time of its usage [12].

After an initial evaluation, the Amazon EC2 solution was selected for several reasons: firstly, it seemed to offer the most suitable trade-off between price and value; additionally, it currently seems to be the most widely spread cloud platform [13]; compared to other platforms (*e.g.* GoGrid), it offers a lower time of resource allocation and release (this affects costs); finally, its instances seem to be more effective than similar systems for sequential operations[13].

B. Architecture design overview

The hybrid cloud architecture in fig. 1 consists of a master node (VM) and three worker nodes (VMs) on the private cloud and a worker node (VM "instance") in the public cloud (Amazon EC2). The private cloud virtual machines (guests) reside on two different hosts (connected to a 100Mbps network). The virtual machines use bridged networking which allows the virtual interfaces to connect to the outside network through the physical interface: as a result, they appear as normal hosts to the rest of the network. On the master node a shared file system's portion is configured, for the purpose of enabling data exchange among all the nodes, using SSHFS[2].

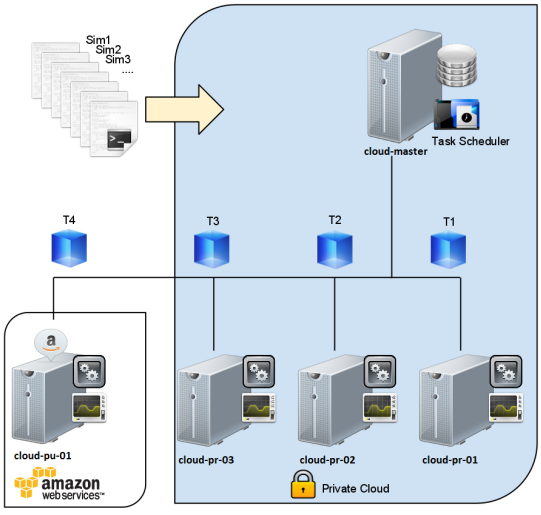

Fig. 1. Hybrid cloud architecture

[1]One EC2 Compute Unit (ECU) provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor

[2]SSHFS is a file-system client based on the SSH File Transfer Protocol

All the nodes have very similar processors but differ in their amount of RAM. In the tested infrastructure all the nodes on the private cloud are full virtualised [14] and use a virtualization solution based on XEN and KVM (Kernel-based Virtual Machine) technology [14].

On the master node there is a database that contains 3 tables: (*i*) *simulation weights*, which associates to each simulation a weight depending on its characteristics (for example, the expected amount RAM memory required by the simulation); *ii*) *machine characteristics*, describing virtual machines in terms of free memory (RAM), processor speed (CPU) and available disk space; *iii*) *state of virtual machine* describing the current state (available, unavailable) of each VM. Also the *Task Scheduler* resides on the master node: it is a java application that builds the tasks (i.e. one or more simulation files), depending on the current policies (*e.g.* free memory of the available worker nodes), and sends them to the corresponding worker nodes. Conversely, in each worker node the are an agent and a system monitor. The agent detects whether there are pending tasks to be executed and launches them; it also sends the database information about the current status of the worker node. Meanwhile, the system monitor detects real-time information about free RAM and updates the database accordingly.

C. Task scheduling process

The process begins when the master node receives Tcl-files containing the details of the simulations to be executed. As previously mentioned, a *task* is a set of *n* simulations sent to a worker node for the purpose of parallel running. Notably, worker nodes compute a task at a time: it is then important to build tasks so as to optimize the exploitation of resources and, at the same time, avoid their overloading. The initial learning phase serves for this purpose: in fact, during this learning phase the weight of each simulation (i.e. its required resources) is detected. More in particular, a table is built to highlight relations between the simulation description files and the allocated memory (RAM). This table is the key for estimating the resources required by any possible simulation.

Given this estimation, the scheduling algorithm builds a task tailored to the available memory (RAM) on a worker node, according to the real capacity of the system. The work scheduling is performed by the *task scheduler*, available on the master node: the scheduler can be considered the brain of the entire cloud infrastructure.

The task scheduler constantly reads the information stored in the database in order to know the resources available throughout the cloud and the queue of simulations waiting for processing. The scheduler adopts the following procedure to build tasks:

1) The master node queries the database to know the worker nodes in "available" state.
2) Database is queried by the task scheduler for the exact value of free memory of each worker node.
3) For each worker node in the available state:
   a) Tcl files are combined so that the sum of the weights does not exceed 90% of the total available memory on the worker node: the expected memory allocation is inferred thanks to the learning phase.
   b) The whole task is sent to the respective worker node, whose agent runs all the simulations.
   c) After the completion of the entire task, the agent reports to the database that the worker node is available for new tasks. All the resources are released and the output files are made available.

The algorithm will loop from step 1 until the queue of files gets empty. In this way, a user can easily compose a set of simulations and run them in the parallel hybrid architecture: the master node will handle the whole job, optimizing the resources, while the worker nodes will compute almost to the saturation of their resources.

Notably, during the execution of simulations (step *b*) a second phase of learning (*self-learning* phase) takes place. The architecture real-time analyzes the consumption of resources (RAM, disk space, elapsed time) by each task and adds corresponding new records in the table of learning, thus refining it. In fact, if a simulation is not described in the learning database, the master node will identify it with a worst one (already stored) in order to keep the system stable.

As anticipated, the scheduler has been optimized for the consumption of memory RAM. The effectiveness of the approach was demonstrated by the comparison to a random scheduling over the same hybrid scheduler architecture: the achieved results confirm (Fig. 2) that the optimization was effective and enabled a decrease in the simulation time. Based on these results the same architecture will be extended, in the future, by further policies (for example CPU performance, economic costs, power consumption, etc.). Other existing tools, like Condor [15], do not offer the features and advantages required for this environment. Moreover, the proposed solution implements more sophisticated algorithms.

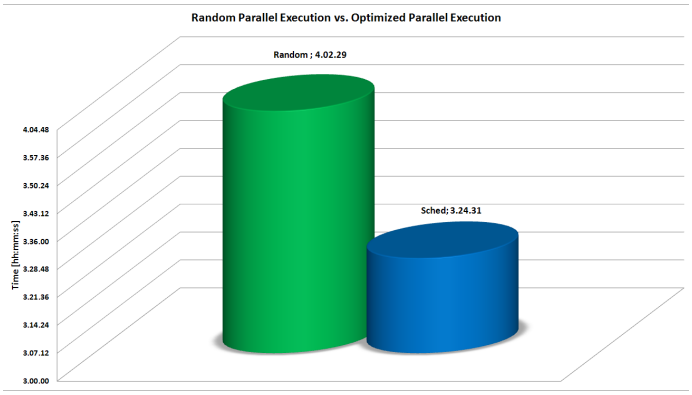

Fig. 2. Validation of the proposed algorithm

IV. Performance Analysis

A. Initial Learning Phase

All in all, the initial learning phase is meant to facilitate the estimation of the computational resources required by each simulation — depending on their respective settings. The estimation is useful to optimize the overall simulation process,

Sim	#Vehicles	Simulation Time [sec.]	Tx Power [dBm]	Pkt rate [Hz]	Elapsed time [hh:mm:ss]	RAM [Mbyte]	Output size [Mbyte]
Sim 1	1000	100	10	15	03:39:20	409.9	24400
Sim 2	1000	100	10	10	03:27:53	386.2	23900
Sim 3	1000	40	7	10	03:40:53	181.8	23990
Sim 4	1000	40	7	5	02:06:37	179.8	12364
Sim 5	750	100	10	15	03:39:07	150.1	23900
Sim 6	750	100	10	10	03:38:13	147.4	23307
Sim 7	750	40	7	10	01:57:09	141.8	14785
Sim 8	750	40	7	5	01:07:46	135.5	7400

TABLE I
SOME SIMULATION RESULTS OF THE LEARNING-STEP

with an appropriate scheduling of simultaneous works; this involves a preliminary definition of metrics addressing the computational load. Based on the authors' experience, the following three *simulation metrics* (SM) have been considered:

1) amount of allocated memory (RAM);
2) simulation time;
3) size of the output file.

These figures are all measured by ad-hoc scripts and vary depending on simulation settings: in fact, different settings are expected to have different impacts on the metrics.

In all the simulated VANET scenarios, the number of vehicles is fixed throughout the simulation, their mobility patterns are *a priori* known, all the mobile nodes generate packets at a same rate and the transmitted power is the same across the network. Consequently, the following high-level parameters have been considered to assess the SMs:

1) number of vehicles;
2) simulated (elapsed) time (sec);
3) packet delivery rate (Hz);
4) transmitted power (dBm).

A large set of simulations (more than 100) have been carried out to accomplish the learning phase. The following settings have been adopted:

- number of vehicles [50, 100, 250, 500, 750, 1000];
- simulation time [40, 100] sec;
- packets delivery rate [5, 10, 15] Hz;
- transmitted power [7, 10] dBm.

The simulations have been sequentially executed over a single machine (worker node), in order to obtain consistent reference values. The outputs of the simulations have been measured by the SMs, getting a series which has been rounded up into discrete values. The discretization in the learning phase is meant to make the measurement simpler: the relevance of this approximation in considered negligible with respect to the expected statistical variations and other hardly measurable dependencies (e.g., on the used hardware). Table I shows the resulting raw values: they facilitate the assignment of appropriate weights to simulations for the task composition.

B. Second testing phase

During the second testing phase 46 simulation files (divided into 20 tasks) were executed in the cloud platform. The details of the virtual machines used for the test are shown in table III. The Amazon instance used for the tests is an EBS Standard Large Instance [m1.large] at a cost of $ 0.38 per hour [12].

Table II shows the simulation time of the tasks performed in *cloud-pr-03*. In the same table are also displayed the number

Task	Simulation files	Time (hh:mm:ss)	Time_seq (hh:mm:ss)
Task 1	4	01:22:44	1:54:25
Task 2	1	01:29:09	1:30:33
Task 3	2	00:52:28	1:42:24
Task 4	2	00:44:36	1:22:10
Task 5	1	00:34:05	0:34:21
Task 6	1	00:33:10	0:33:28

TABLE II
SIMULATION TIME ON CLOUD-PR-03

of simulation files for each task performed and their respective simulation times, which are also plotted in Fig. 3, both for the parallel and sequential executions. A decrease in simulation time can be noticed when the tasks (Task 1,3,4) are composed of multiple simulation files, while this time remains almost unchanged when the tasks are composed of a single simulation file (Task 2,5,6). The scheduler generates a one-simulation task when the memory available μ on a worker node is sufficient to run a simulation whose weight ρ_1 has a value close to μ. The scheduler can not add another simulation to the task when there are no simulations satisfying the condition

$$\rho_2 < \mu - \rho_1$$

where ρ_1, ρ_2 are the weights of the first and second simulation and μ is the available RAM on the worker node.

The total simulation time, the number of tasks and simulation files executed in each worker node and the total simulation time of all the 46 simulations performed sequentially in cloud-pr-03 are summarized on in table IV.

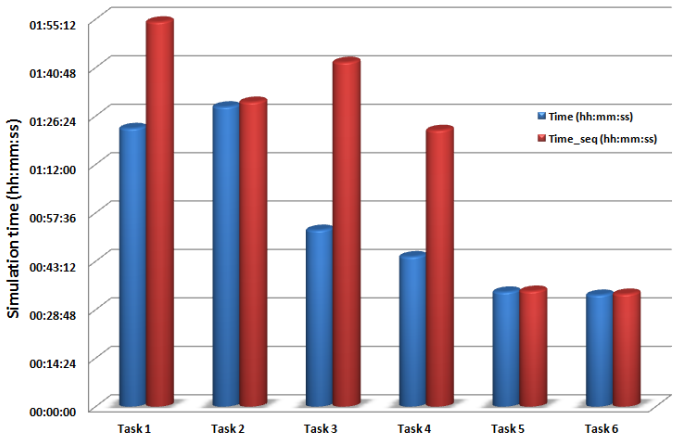

Fig. 3. Simulation time on cloud-pr-03

The values in table IV have been plotted in Fig. 4: the figure shows that the time required for the execution of 20 tasks (46 simulation files) in the cloud platform is significantly lower

VMs	OS	CPU model	Virtual CPU	RAM (GB)	Kernel
cloud-master	Debian Squeeze 6.0.2	Intel Xeon 5140 @ 2.33 GHz	2	1	Linux 2.6.32-5-xen-amd64
cloud-pr-01	Debian Squeeze 6.0.3	Intel Xeon 5140 @ 2.33 GHz	2	3	Linux 2.6.32-5-xen-amd64
cloud-pr-02	Debian Squeeze 6.0.3	Intel Xeon 5140 @ 2.33 GHz	2	2	Linux 2.6.32-5-xen-amd64
cloud-pr-03	Debian Squeeze 6.0.3	Intel Xeon X5660 @ 2.80 GHz	2	7.5	Linux 2.6.32-5-amd64
cloud-pu-01	Debian Squeeze 6.0.1	Intel Xeon E5507 @ 2.27 GHz	2	7.5	Linux 2.6.32-5-xen-amd64

TABLE III
CLOUD VIRTUAL MACHINES SPECIFICATIONS

VMs	Tasks executed	Simulation files	Time(hh:mm:ss)
cloud-pr-01	3	10	00:59:30
cloud-pr-02	9	16	00:52:30
cloud-pr-03	6	11	05:36:12
cloud-pu-01	2	9	01:39:37
cloud-pr-03-seq	-	46	11:27:53

TABLE IV
TOTAL SIMULATION TIME

than the time required to perform the same task sequentially. Figure 3 shows that, although the cloud-pr-03 is the most powerful virtual machine of the cloud platform, only 6 tasks (11 simulation files) have been assigned to it, with a simulation time of 05:36:12; instead 9 tasks (16 simulation files) were assigned to cloud-pr-02, with a simulation time of 00:52:30.

Since cloud-pr-03 is the most powerful machine in the cloud platform, tasks that require a high amount of memory (RAM) were preferably assigned to it, resulting simulation files with greater weights, leading to longer execution time. In Fig. 4, the column corresponding to cloud-pr-03-seq represents the total simulation time for the sequential execution, during the learning phase. This permits to compare the execution performance achieved in the first sequential phase (without using cloud infrastructure), and the execution over the cloud infrastructure: the latter led to simulation time which is the 48% of the former.

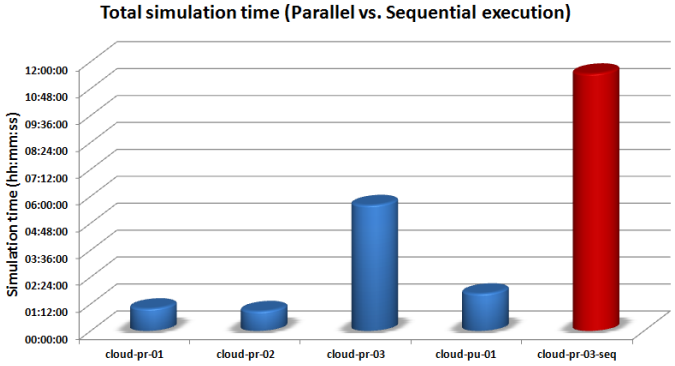

Fig. 4. Total simulation time

V. FUTURE WORKS & CONCLUSIONS

In this paper a new architecture, based on a virtualised cloud-computing environment, has been presented. The architecture is aimed at improving the performance of the heavy computation required by VANET simulations. The proposed solution is a cross-platform cloud architecture made up by virtual machines. The system is based on the model of hybrid cloud computing and is able to perform a large number of simulations in parallel, so as to reduce the total simulation time. The core of the architecture is a scheduler that is used to assign simulations to VMs after grouping them into tasks, depending on available resources (memory) on each node. In this research, the proposed architecture is optimized for the NS-2 software. However, this solution is not bound with a specific tool, but it can support any other simulation software, like NS-3. In the research perspective, the impact seems promising. In fact, results demonstrate that a deep resource optimization is achieved, while the execution time and costs of simulations gets significantly reduced. On the other hand, the envisioned solution poses new issues requiring new solutions and deeper insights. For instance, a future study will develop a Graphical User Interface to automatize all the operations; the GUI will also automatically manage the configuration of the files required to set simulation parameters. Future works will also investigate the effect of the additional parameters, other than RAM, to be used for the optimization of scheduling (e.g., the number of available virtual cores or the free disk space).

REFERENCES

[1] E.Weingartner, H. vom Lehn, and K.Wehrle A performance comparison of recent network simulators, In *Proc. of the IEEE International Conference on Communications*, 2009.

[2] E.Weingartner, H.vom Lehn and K.Wehrle, A performance comparison of recent network simulators, *Proc. of the IEEE International Conference on Communications*, 2009.

[3] IEEE Standard 802.11p, Wireless Access in Vehicular Environments, July 2010.

[4] Z.Shuai, C.Xuebin, Z.Shufen and H.Xiuzhen, The comparison between cloud computing and grid computing, *Proc. of the IEEE International Conference on Computer Application and System Modeling*, 2010.

[5] P.Alexander, D.Haley, and A.Grant, Cooperative Intelligent Transport Systems: 5.9-GHz Field Trials, *In Proceedings of the IEEE Journal*, Vol. 99, N. 7, July 2011.

[6] iTetris an Integrated Wireless and Traffic Platform for Real-Time Road Traffic Management Solutions, [Online] Available: http://ict-itetris.eu/.

[7] M. Kozlovszky, A. Balaskó and A. Varga Enabling OMNeT++-based simulations on Grid Systems, in *OMNeT++ 2009: Proceedings of the 2nd International Workshop on OMNeT++ (hosted by SIMUTools 2009)*.

[8] P. Mell, T. Grance The NIST definition of Cloud Computing [Online] Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf, 2012.

[9] Q.Chen, F.Schmidt-Eisenlohr, D.Jiang, M.Torrent-Moreno, L.Delgrossi, and H.Hartenstein, Overhaul of ieee 802.11 modeling and simulation in NS-2, *In Proc. of the 10th ACM MSWiM*, 2007.

[10] Amazon, Inc., Amazon Elastic Compute Cloud, User Guide, API Version 2011-12-15, [Online] Available: http://awsdocs.s3.amazonaws.com/EC2/latest/ec2-ug.pdf, 2012.

[11] Amazon, Inc., Amazon Elastic Block Store (EBS), [Online] Available: http://aws.amazon.com/ebs/, 2012.

[12] Amazon, Inc., Amazon Elastic Compute Cloud (Amazon EC2), [Online] Available: http://aws.amazon.com/ec2/, Dec. 2008.

[13] A. Iosup, S. Ostermann, N. Yigitbasi, R. Prodan, Th. Fahringer, and D. Epema Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing *Proc. of the IEEE Transactions on parallel and distributed systems*, vol. 22, No.6, pp. 931-945, June 2011.

[14] A. Chierici and R. Veraldi A quantitative comparison between xen and kvm, *Proc. of 17th International Conference on Computing in High Energu and Nuclear Physics*: Conference Series 219 (2010).

[15] Douglas Thain, Todd Tannenbaum, and Miron Livny Condor and Grid in Ian Foster and Carl Kesselman, editors, The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann, 2003, 2nd edition.