CloudNetSim - Simulation of Real-Time Cloud Computing Applications

Tommaso Cucinotta, Aram Santogidis
Bell Laboratories, Alcatel-Lucent, Dublin

In this paper, we describe CloudNetSim, a project aiming to realise a simulation platform supporting our ongoing and planned research activities in the area of resource management and scheduling for distributed QoS-sensitive and soft real-time applications. It is based on OMNeT++, integrating in the platform a set of modules for the simulation of CPU scheduling, including hierarchical scheduling at both levels of the hypervisor and guest Operating System, as needed when simulating cloud infrastructures. Thanks to the modularity of OMNeT++, CloudNetSim may easily leverage many existing simulation models already available for networking, including standard network components and protocols, such as TCP/IP. After a brief overview of related simulation tools found in the literature, and the discussion of their limitations, we provide a detailed description of the internals of our simulator. Then, we show results gathered from a few representative scenarios demonstrating how its behaviour matches with the one of simple real applications.

1 Introduction

Cloud Computing is gaining momentum as one of the key innovations disrupting the world of computing, constituting a page turn from the old ages of Personal Computing to a new era of massively distributed cloud applications and services accessed from a plethora of devices with increased mobility support. Cloud Computing is also generating a continuous pressure towards the research community, for introducing innovations and novel mechanisms promising to support better the nowadays and future computing scenarios. As connectivity evolves towards higher bandwidth and lower latency, more and more soft real-time (RT) and interactive on-line applications are becoming increasingly used and popular [17]. These include many on-line interactive cloud applications, such as office suites (e.g., Google Docs) or e-Learning platforms [7], virtual desktop, and online massively parallel games.

When working at the lowest layers of the cloud infrastructure, and specifically at the hypervisor, Operating System (OS), and CPU scheduling levels, it is often difficult if not impossible to gain access to realistic test-beds over which to carry out research activities in the field. Often, it is very handy and convenient to have available tools that may assist researchers in simulating cloud deployments and end-to-end distributed applications, with a sufficient level of abstraction depending on the research purposes and scope.

However, simulation of distributed soft real-time applications over general-purpose computing platforms and networks is troublesome due to the lack of proper tools. Various simulators exist in the areas of networked systems and real-time systems, and recently a few simulation tools have become available in the area of Cloud Computing. In general, the existing tools were lacking the fundamental ability to integrate the various cross-domain simulation aspects that affect the end-to-end performance (see Section 2).

In this paper, we introduce CloudNetSim, a project aiming to provide a simulation platform to assist the experimentation with resource management and scheduling in cloud computing. At a glance, its main features comprise: packet-level simulation of end-to-end network communications between clients and servers distributed throughout a cloud infrastructure; simulation of computing resources including but not limited to CPU scheduling both at the hypervisor and at the guest OS levels; support for virtual machine (VM) deployment strategies; modularity and extensibility, with the possibility to introduce additional scheduling policies, VM deployment strategies and application models as needed. We aim to keep an abstraction level that allows for simulation of thousands of nodes and applications, gathering the necessary QoS metrics, within an affordable time.

Even though CloudNetSim targets simulation of cloud applications, the presented work may also be used for simulating networked soft real-time and embedded systems.

2 Related Work

In this section, the simulation tools mostly related to the presented work are briefly introduced. They fall roughly in the categories of real-time systems simulators, network protocols simulators and cloud computing simulators.

In the area of RT and embedded systems, many simulation tools deal with simulation of CPU scheduling, including *RTSim* [18], *MAST* [10], *MAST2* [9], *SimTrOS* [19]

and others [1], just to mention a few. However, either they are exclusively focused on hard RT and embedded systems and they do not support general-purpose schedulers and related technologies, or they neglect entirely the networking aspects. Some tools integrate simulation of CPU scheduling and technologies/protocols for CAN busses or Wireless Sensor Networks [6], however these tools are hardly reusable in the context of general purpose technologies.

In the area of networking and distributed systems, many tools provide an accurate simulation of networking technologies and packet-level simulation of networking protocols [21, 11, 20]. For example, *NS2*[1] is probably one of the most widely known open-source simulators used in research about network protocols, whose development started in 1996. It supports packet-level simulation of many Internet protocols and technologies, including TCP/IP and wireless networks. However, the simulator derives from a quite old code base, where functionality has been evolving over years, split around C/C++ and Object Tcl code. This resulted in the lack of modularity and clean interfaces for extending its functionality. It is not a case that, from 2004, a new *NS3*[2] project was born with the intention of a complete redesign of the internals of the simulator, and ultimately dropping compatibility with NS2. Unfortunately, this resulted in a set of features not (yet) as complete as in NS2. A completely alternative project is the open-source *OMNeT++*[3], free for academic use, and commercially licensed as *OMNEST*[4]. OMNeT++/OMNEST is a simulation platform with a completely modular design where generic modules can be connected in arbitrarily complex topologies and communicate with each other, all integrated with an Eclipse-based development environment including a visual topology editor. One of the main uses of OMNeT++ is in connection with the INET Framework[5], an open-source communication networks simulation package including a set of modules for simulation of Internet technologies and protocols, including TCP/IP, IPv6, Ethernet, PPP, 802.11, MPLS, and others.

However, all of these network simulators simply do not include any CPU scheduling infrastructure. In a cloud environment, where multiple VMs may be multiplexed on the same physical host, processor and core, it is important to simulate CPU scheduling, to get a comprehensive picture of the end-to-end response-time and performance. Especially when dealing with low-latency cloud applications deployed in future scenarios with fine-grained cloud data centres, tools are needed to support a *comprehensive* and *integrated* simulation of multiple resources, such as CPU, network and storage, that allow for modelling distributed applications, particularly those with QoS requirements, developed in the context of general-purpose technologies.

Recently, a few simulation tools have become available [2, 14, 4, 12] targeting the specific simulation needs arising in the area of Cloud Computing. *CloudSim* [2] is a Java-based simulation platform modelling various aspects of cloud computing infrastructures such as high-level simulation of data centres with virtualized hosts, energy consumption models and federated clouds. Versions prior to 2.0 have a very simple networking model at the flow level, with statically configured latency and bandwidth values among locations, while from version 2.0 a better network simulation functionality was added.

CloudSim is derived from *GridSim* [3], thus its architecture is still strongly tied to the modelling and simulation of GRID scenarios, with a focus on load balancing within the data centre, rather than gathering performance metrics over end-to-end deployments of general-purpose cloud computing applications, as in our proposed CloudNetSim.

iCanCloud [14, 4] is a simulator platform for cloud computing based on OMNeT++, with the capability to configure various resource management policies for the hypervisor, virtual machine models aiming to simulate the behaviour of real world CPUs, data centre topologies that mimic the architecture of state of the art cloud computing infrastructures (e.g. Amazon EC2[6]) and data storage emulation. Still, this tool is lacking the essential capability to simulate the variety of heterogeneous networks involved in the end-to-end cloud service supply chain. However, being based on OMNeT++ as our framework, iCanCloud has interesting modules that we might re-use, such as the storage models inherited from *SIMCAN* [16, 15], a simulator of local and remote storage systems, including NFS and parallel file systems.

GreenCloud [12] is an NS2-based C++ simulator aiming to model the energy consumption of data center IT equipment (e.g. computers, network switches and communication links), to help the design of energy efficient architectures. However, GreenCloud needs merely a rough estimate of the expected computing workload on the nodes, for its power consumption estimates.

Overall, some of the mentioned simulators targeting cloud computing focus specifically on aspects of the infrastructure related to computing within the data centre, neglecting the important aspects of communications over the Internet or the access network. Others try to enrich an accurate simulation of the network by adding rough computing models which cannot capture a similar level of detail, when addressing QoS and responsiveness. However, consideration of the whole end-to-end chain is very important for the overall QoS delivered to remote customers/users.

As a consequence, we could not find in existing tools

[1] More information at: `http://www.isi.edu/nsnam/ns/`.
[2] More information is available at: `http://www.nsnam.org/`.
[3] More information is available at: `http://omnetpp.org/`.
[4] More information is available at: `http://www.omnest.com`.
[5] More information at: `http://inet.omnetpp.org/index.php`.
[6] More information is available at: `aws.amazon.com/ec2/`.

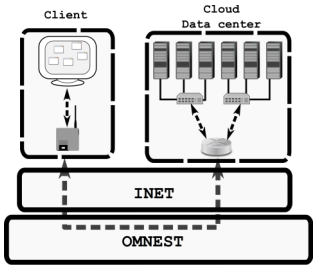


Figure 1. High-level software architecture

features properly matching with the particular needs of research on the topics of resource management and scheduling for interactive, real-time and low-latency cloud and distributed applications.

3 Proposed Approach

One of the primary goals of the overall ongoing CloudNetSim project is to integrate within a single simulation platform the major factors contributing to end-to-end latency of low-latency cloud applications, namely networking, computing and disk access, including overheads due to virtualization (both machine and network virtualization).

We opted to implement the computing part of the simulation on top of OMNEST (see Figure 1), due to its relative maturity, modular design and extensibility. We realized a set of OMNEST modules in order to model computing and CPU scheduling within physical hosts and VMs, as happening within a cloud computing data centre; these inter-mix with the already available network communication modules, resulting in a more comprehensive emulation of the major contributions to end-to-end response-times. At this preliminary stage, disk access has been greatly simplified, but we plan to consider it more thoughtfully later.

Simulating large infrastructures with such a fine-grained level of detail for computing and networking resources may present performance and scalability challenges. However, it has been shown [13] that parallelisation techniques can be effectively applied to OMNeT++ simulations in a seamless fashion, without requiring changes in the code. These will certainly be useful for our planned future investigations.

Overall Design. The core component for modelling computing elements is *CloudNode*. It is built on top of the *NodeBase* compound module of INET which models a network host, and provides interface and network layer functionality. CloudNode additionally incorporates a number of modules that emulate the computing part of the module (see Figure 2), i.e., the CPU Scheduler, modules for applications and for data storage emulation. Moreover, the CloudNode

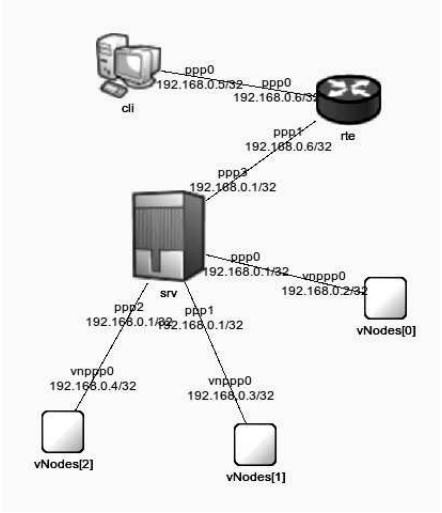


Figure 2. OMNEST representation of a simple topology.

modules can be interconnected with each other in a hierarchical fashion, effectively modelling VMs running within a physical host. Figure 2 illustrates the topology of a client connected through a router to a host running 3 VMs.

In OMNEST terminology, CloudNode is a *compound module* that extends NodeBase (see Figure 3 for an overview of its inner design). It is an aggregation of *simple modules* that allow for modelling various aspects of the software stack typical of virtualized infrastructures. As depicted in Figure 3, CloudNode includes simulation of network capabilities as inherited from NodeBase, data storage and CPU scheduling. The networking capabilities have been customised by adding *SchedPPP*, a module extending the INET PPP interface which is controllable from the CPU scheduler. This is necessary in order to "suspend" the network connectivity of a VM, when it is preempted from execution by the CPU scheduler. A similar *SchedEth* module has been realised for Ethernet.

The *Scheduler* within a CloudNode is able to schedule an arbitrary number of *Schedulable* entities over a configurable number of available CPUs. Also, these can be connected to a *data storage model* in order to model suspension on I/O. Interestingly, a CloudNode is schedulable on its own. This allows VMs to be modelled as CloudNode instances connected to the Scheduler of the outer CloudNode representing the host they are deployed within.

Scheduler Design. Schedulable entities represent software running in the system, including both applications or components at the hypervisor level, and those within guest VMs. The *Schedulable* interface permits the Scheduler to

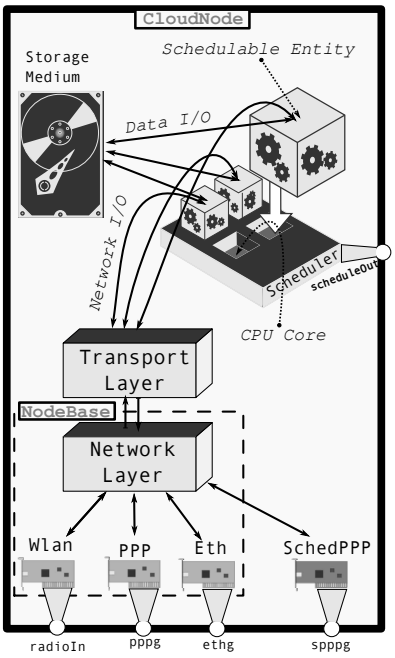


Figure 3. CloudNode design.

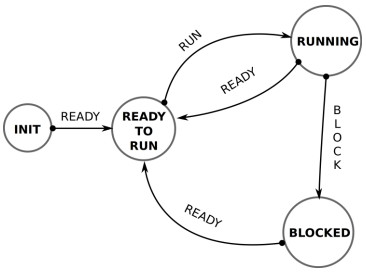


Figure 4. Schedulable entities FSM.

manage their execution state. All these entities extend the *BaseSchedulable* class that implements the well-known Finite State Machine (FSM) in Figure 4. The logical communications between the Scheduler and its managed entities, necessary to realise the mentioned FSM behaviour, is conducted through the *scheduleIn/scheduleOut* ports of the Schedulable interface, by exchanging custom defined OMNEST messages. Specifically, the Scheduler notifies ready-to-run entities whenever a CPU is assigned or revoked to them, according to the scheduling algorithm in use within the Scheduler. The entities, on their own, notify the Scheduler whenever they need to suspend for data I/O, networking, or timer operations.

The CPU is modelled in the scheduler with a few configurable parameters controlling its power-saving capabilities, including the frequency at which it is running, and whether it is in a deep-idle state. Therefore, messages from the Scheduler to the entities also include the frequency change information, needed to allow applications to modulate their execution time behaviours accordingly. This allows for simulation of multi-processor and multi-core hosts with CPU power-saving capabilities. However, an exact strategy to switch among the available CPU frequencies (i.e., mimicking the behaviour of the `cpufreq` governors in Linux) is still work in progress. Also, we only modelled a single idle-state of the CPU with a configurable wake-up latency, as at the moment there is no interest in modelling the multitude of idle states in modern CPUs.

We realised 3 scheduling algorithms: Fixed Priority (FP), Round-Robin, Linux Completely Fair Scheduler (CFS). These can be hierarchically composed with each other. This is shown through the example in Figure 5.(a), where a typical Linux set-up is shown with 6 applications running under various scheduling policies, as detailed in Figure 5.(b). With the proposed architecture, multiple real-time tasks at the same priority under the POSIX SCHED_RR policy are represented as connected to an instance of the Round-Robin Scheduler, which in turn is connected to the FP Scheduler at the needed priority level. Also, SCHED_OTHER tasks are connected to a CFS Scheduler connected to the FP Scheduler at priority 0.

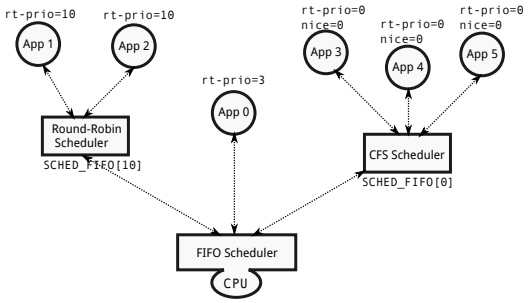
Configuration of the Scheduler(s) topology is simplified by specifying for each application the desired scheduling parameters (including the nice level, in case of SCHED_OTHER entities), and the CloudNode instantiates the required Scheduler modules and interconnections as needed. Note that the overall Scheduler design allows for an easy introduction of new algorithms.

Application Model. Applications are modelled in CloudNetSim as Schedulable entities, executing sequentially a list of instructions. Following the steps of RTSim [18], the purpose of the simulation is not functional simulation, but rather performance evaluation. Therefore, allowed instructions are for now: computing for a fixed amount of time (scaled linearly with the CPU frequency); wait for the transfer of a fixed number of blocks to/from the storage medium; change dynamically the scheduling parameters of the application. Also, a few instructions are being realised allowing for modelling (the impact on performance of) communications among various parts of a distributed cloud application.

A convenience scripting syntax has been defined, so that simple application models may easily be provided through text-based input files to the simulation.

4 Calibration and Simulation Experiments

In this section we report results from a few experiments we ran in order to show how the parameters of the simulated models may be calibrated so that its outcome matches with the behaviour measured from a real simple scenario.



(a)

App	Priority	Policy
0	3	SCHED_FIFO
1	10	SCHED_RR
2	10	SCHED_RR
3	0	SCHED_OTHER
4	0	SCHED_OTHER
5	0	SCHED_OTHER

(b)

Figure 5. Hierarchical scheduling of processes based on policy

	Real world	Simulation
Host (idle)	0.384 +/- 0.040 ms	0.388 ms
Host (hog)	0.322 +/- 0.034 ms	0.333 ms
VM1 (idle)	0.482 +/- 0.034 ms	0.462 ms
VM1 (hog)	0.377 ms +/- 0.036 ms	0.399 ms

Table 1. Ping times statistics for the real-world (left) and simulated (right) scenarios.

Ping Test. We consider a simple topology with a physical host running two VMs connected through a network router to a client that pings the physical host and the VMs (see Figure 2). After calibration of the simulation model parameters, its results are compared with numbers obtained by the corresponding real-world example. In the latter, we used an Intel i5-2520M @ 2.50GHz laptop client pinging a Linux machine equipped with an Intel Xeon E5-2687W CPU whose frequency was fixed at 3.1 GHz, and in which all cores except one have been put offline, and hyper-threading has been disabled, to create the simple scenario reproduced in simulation. Also, a guest KVM Linux OS has been run on the server machine. The host and the VM have been continuously pinged for one minute every half second, when the host was idle, and when it was loaded. The obtained ping times average and standard deviation are shown in Table 1, in both real-world and simulated cases.

The overall ping latency towards the host is the result of summing up delay contributions due to network delay to reach the server, CPU wake-up from idle,networking stack execution for replying to the ping, then back to the client. When pinging the VM, further contributions are due to the context switch to schedule the VM and guest OS networking stack execution for replying to the ping.

CFS Test. We ran another ping experiment using the CFS as the hypervisor scheduler. We verified that, despite a 2nd VM hogging the CPU, the pinged idle VM was responding immediately to the ping, preempting the other one. This behaviour is in sync with the CFS algorithm since the pinged VM, waking up from a blocked state, runs immediately, since its virtual run-time is much lower than the one of the CPU-bound VM continuously executing.

Then, to verify the behaviour of the CFS in presence of different nice values, we considered another simple scenario with three applications running CPU bound tasks on a host and we compared the obtained simulated versus real figures.

We use a `load.sh` shell script realising a simple `for` loop for the number of iterations provided as argument. When running on an Intel i5-2520M CPU at fixed 2.50 GHz frequency, `load.sh` takes 1 second to complete with an argument of 177000. In single-core mode, we run three tasks with the default nice value (0), however the third one is reniced to (10) at half execution. This is obtained as:

```
time ./load.sh 177000 &
time ./load.sh 177000 &
time ./load.sh 88500
time nice ./load.sh 88500
```

The obtained results show that the first two processes completed in less than 2.6 seconds, whilst the reniced process completed after $1.56 + 1.49 = 3.05$ seconds:

```
0.47u 0.02s 1.56r   ./load.sh 88500
0.93u 0.03s 2.55r   ./load.sh 177000
0.95u 0.03s 2.58r   ./load.sh 177000
0.51u 0.00s 1.49r   nice ./load.sh 88500
```

The same experiment has been arranged in the **simulated** model, using the renice instruction explained in the previous section for changing dynamically the third process nice level at half of its execution. This resulted in the following output, gathered from the OMNeT++ logs:

```
T=3.004008  TestCloudNode.srv.tcpApp[0]
T=2.560008  TestCloudNode.srv.tcpApp[1]
T=2.554008  TestCloudNode.srv.tcpApp[2]
```

These results validate the correct behaviour of the CFS Scheduler model, in the mentioned scenario.

5 Conclusions and Future Work

In this paper we presented CloudNetSim, a simulation platform suitable for capturing the behaviour of end-to-end

time-sensitive and particularly low-latency distributed applications. The platform exploits the native OMNEST and INET capabilities for network simulation, integrating simulation of computing and storage access in virtualized environments. We plan to use this platform for our ongoing and planned research in the area of resource management and scheduling for soft real-time cloud computing applications. The presented simulation models are very important to simulate the impact on performance of sharing physical computing resources within the infrastructure, as often done by cloud providers trying to achieve high consolidation levels. However, CloudNetSim may also be useful for simulation of soft real-time distributed embedded systems.

The presented work may be extended along various lines of action: the CPU scheduling models may be refined by adding further scheduling policies, e.g., one mimicking the Xen scheduler [5, 8]; the storage access model is very simple, but re-usable modules from other projects such as SIMCAN might be integrated; the performance achievable with the integrated multi-resource simulation on large scale systems has to be checked, an area where parallelisation techniques such as [13] might be useful.

References

[1] M. Ashjaei, M. Behnam, and T. Nolte. The design and implementation of a simulator for switched ethernet networks. In *Proc. of the 3rd International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems*, pages 57–62, Pisa, Italy, July 2012.

[2] R. Calheiros et al. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50, 2011.

[3] R. N. Calheiros, R. Ranjan, C. A. F. D. Rose, and R. Buyya. Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services. *CoRR*, abs/0903.2525, 2009.

[4] G. Castane, A. Núñez, and J. Carretero. iCanCloud: A brief architecture overview. In *Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th International Symposium on*, pages 853–854, 2012.

[5] L. Cherkasova, D. Gupta, and A. Vahdat. Comparison of the three cpu schedulers in xen. *SIGMETRICS Perform. Eval. Rev.*, 35(2):42–51, Sept. 2007.

[6] M. Chitnis et al. Impact Of The Operating System on the QoS offered by an IEEE 802.15.4-compliant Sensor Network. In *Proceedings of the 7th IFAC International Conference on Fieldbuses and networks in industrial and embedded systems*, Tolouse, France, Nov 2007.

[7] T. Cucinotta et al. Virtualised e-learning with real-time guarantees on the irmos platform. In *Proceedings of the IEEE International Conference on Service-Oriented Computing and Applications (SOCA 2010)*, pages 1–8, Perth, Australia, December 2010.

[8] G. Dunlap. Scheduler development update. Xen Summit Asia 2009, Shanghai, 11 2009.

[9] M. G. Harbour et al. Modeling real-time networks with mast2. In *Proc. of the 2nd International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems*, pages 51–56, Porto, Portugal, July 2011.

[10] M. G. Harbour, J. J. G. García, J. C. P. Gutiérrez, and J. M. D. Moyano. Mast: Modeling and analysis suite for real time applications. In *Proceedings of the 13th Euromicro Conference on Real-Time Systems*, ECRTS '01, pages 125–, Washington, DC, USA, 2001. IEEE Computer Society.

[11] T. Issariyakul and E. Hossain. *Introduction to Network Simulator NS2*. Springer, 2009.

[12] D. Kliazovich, P. Bouvry, and S. U. Khan. A packet-level simulator of energy- aware cloud computing data centers. *Journal of Supercomputing*, 62(3):1263–1283, 2012.

[13] D. Lugones, K. Katrinis, M. Collier, and G. Theodoropoulos. Parallel simulation models for the evaluation of future large-scale datacenter networks. In *Proc. of the 2012 IEEE/ACM 16th International Symposium on Distributed Simulation and Real Time Applications*, DS-RT '12, pages 85–92, Washington, DC, USA, 2012.

[14] A. Núñez, J. L. Vázquez-Poletti, A. C. Caminero, G. G. Castañé, J. Carretero, and I. M. Llorente. iCanCloud: A flexible and scalable cloud infrastructure simulator. *J. Grid Comput.*, 10(1):185–209, Mar. 2012.

[15] A. Nunez et al. Design of a flexible and scalable hypervisor module for simulating cloud computing environments. In *Performance Evaluation of Computer Telecommunication Systems, International Symp. on*, pages 265–270, 2011.

[16] A. Nunez, J. Fernandez, J. Garcia, and J. Carretero. New techniques for simulating high performance MPI applications on large storage networks. In *Cluster Computing, 2008 IEEE International Conference on*, pages 444–452, 2008.

[17] E. Oliveros, A. Mazzetti, W. Huther, and A. Menychtas. Irmos deliverable d2.1.3 - final version of requirements analysis report. Technical report, IRMOS Consortium, Nov 2010.

[18] L. Palopoli, G. Lipari, G. Lamastra, L. Abeni, G. Bolognini, and P. Ancilotti. An object-oriented tool for simulating distributed real-time control systems. *Softw. Pract. Exper.*, 32(9):907–932, July 2002.

[19] J. Schneider, M. Bohn, and C. Eltges. SimTrOS: A Heterogenous Abstraction Level Simulator for Multicore Synchronization in Real-Time Systems. In *Proc. of the 2nd International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems*, pages 39–44, Porto, Portugal, July 2011.

[20] A. Varga and R. Hornig. An overview of the omnet++ simulation environment. In *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, Simutools '08, pages 60:1–60:10, Brussels, Belgium, 2008. ICST.

[21] E. Weingartner, H. vom Lehn, and K. Wehrle. A performance comparison of recent network simulators. In *Proceedings of the IEEE International Conference on Communications 2009 (ICC 2009)*, Dresden, Germany, 2009. IEEE.