An Adaptive Load Balancing Middleware for Distributed Simulation

Luciano Bononi, Michele Bracuto, Gabriele D'Angelo, and Lorenzo Donatiello

Dipartimento di Scienze dell'Informazione, Università degli Studi di Bologna
Mura Anteo Zamboni 7, 40126, Bologna, Italy
{bononi, bracuto, gdangelo, donat}@cs.unibo.it

Abstract. The simulation is useful to support the design and performance evaluation of complex systems, possibly composed by a massive number of interacting entities. For this reason, the simulation of such systems may need aggregate computation and memory resources obtained by clusters of parallel and distributed execution units. Shared computer clusters composed of available Commercial-Off-the-Shelf hardware are preferable to dedicated systems, mainly for cost reasons. The performance of distributed simulations is influenced by the heterogeneity of execution units and by their respective CPU load in background. Adaptive load balancing mechanisms could improve the resources utilization and the simulation process execution, by dynamically tuning the simulation load with an eye to the synchronization and communication overheads reduction. In this work it will be presented the GAIA+ framework: a new load balancing mechanism for distributed simulation. The framework has been evaluated by performing testbed simulations of a wireless ad hoc network model. Results confirm the effectiveness of the proposed solutions.

1 Introduction

The simulation is useful to support the design and performance evaluation of many complex systems of interest, often composed by a massive number of interacting entities. For a significant performance evaluation, many complex systems would require the implementation of fine-grained models able to include many factors and to capture many causal effects, at many layers of abstraction. Simulation techniques support layered model composition, arbitrarily complex models and fine-grained details [10]. Limitations are mainly given by the long time required to complete the simulation processes, and by the resource constraints of the execution architectures (like computation-power and memory). In the classical approach, computer simulation is monolithic, that is, a single process execution manages one simulation and mimics the evolution of the model state variables. Given the limited amount of memory resources to represent the model data structures, and the limited computation power that can be provided by a single execution unit, the simulation model scalability may be significantly constrained under the monolithic approach. In addition, the time required to complete a simulation analysis over a single Physical Execution Unit (PEU) could be long enough to make less practical the analysis. To deal with these limitations, an alternative approach is based on the Parallel and Distributed Simulation (PADS).

In PADS, the model execution is supported by many interacting processes, possibly executed in concurrent way over multiple PEUs, and usually referred to as Logical Processes (LPs) [1]. One or more LPs can be executed over different PEUs. The simulation is obtained as the coordinated, concurrent, distributed execution of LPs. In general, frequent synchronizations are required between computation steps of distributed LPs, to ensure a correct simulation execution. To summarize, with PADS it is possible to use an arbitrary number of PEUs, by aggregating the available memory and the computation power resources of shared clusters. This increases the model scalability supported by the simulation architecture. On the other hand, new bottlenecks are originated under the distributed coordination, synchronization, and communication viewpoints [6]. A typical cluster-based execution architecture for distributed simulation can be a dedicated cluster with homogeneous units, or a cluster of heterogeneous PEUs, connected by a computer network. Heterogeneity is intended here in terms of PEUs' performance characteristics, available resources, and background load. In shared clusters, the effect of load given by unpredictable background-processes may drastically influence the simulation execution performance. This can be avoided in High Performance Computing (HPC) clusters, that can be reserved, resulting more effective than shared clusters in their resource/performance ratio. On the other hand, shared clusters are cost-effective solutions, because they are made by commercial off-the-shelf (COTS) hardware, possibly deployed for students labs, personal workstations and other computing facilities. In this case the execution units can be really heterogeneous, the CPUs can range from entry level up to bleeding edge models, and the communication network can be a high speed LAN up to the Internet. Being shared between a community of users, it is realistic to assume the presence of variable background load, both for the CPUs and for the interconnection networks.

Typically, in PADS each LP manages a subset of the Simulated Model Entities (SME). Intuitively, a SME is a single model component (object) of the simulation model characterized by a state (data structure) and a behavior (methods). The amount of SMEs managed by a LP has some direct relationship with the computation time required between two successive synchronizations involving LPs. To realize load balancing, the number of SMEs allocated over LPs, should depend on the available resources of the PEUs where LPs are executed. A static allocation of the SMEs realized without considering the dynamic SME interactions, and the dynamic background computation and communication loads, could result in dynamic unbalancing of LP executions. The time required to complete the simulation runs, usually referred to as the Wall-Clock-Time (WCT), is the metric used for the analysis of the simulation speed. The computation load can be balanced between PEUs, by migrating SMEs from an overloaded PEU to an underloaded PEU. The way to dynamically balance the distribution of computation over the available PEUs, should depend on realtime feedback of current load. This paper proposes and analyzes a distributed mechanism for load balancing based on the realtime adaptive migration of SMEs between LPs in a PADS framework. The preliminary Generic Adaptive Interaction Architecture (GAIA) proposed in [3] has been enhanced in this work, originating the new GAIA+ framework. GAIA+ takes simultaneously into account two main problems of distributed simulation: the computation and communication load balancing issues and the reduction of the communication overheads required to implement the simulation. The two problems are strictly correlated and should not be addressed independently, in order to achieve consistent advantages and results. In addition, GAIA+ introduces heuristics for supporting dynamic load-balancing in simulations over COTS shared cluster systems, characterized by heterogeneous PEUs, and unpredictable background computation and communication loads.

The paper structure is the following: in Section 2 some background concepts and related works about load balancing and the distributed simulation are introduced; in Section 3 we introduce a PADS framework that is adopted as the basis for designing and implementing the GAIA+ load balancing solution; in Section 4 we report some results obtained by the GAIA+ framework adopted for a testbed wireless ad hoc system simulation. Finally Section 5 reports our conclusions and future work.

2 Background and Related Work

The load balancing of parallel and distributed computations has been widely investigated and evaluated. In the simulation field, specific solutions have been proposed to distribute the simulation workload over the processors while reducing the synchronization overhead. In most cases some knowledge is assumed or inferred at compile time about the system and model workload parameters [11]. Some works deal with the opposite assumption: in [6] a process migration mechanism is presented, that reduces the WCT of a parallel simulation. The simulation approach considered is based on an optimized version of the conservative Chandy-Misra synchronization scheme. The proposed mechanism is dynamic and partially distributed. In [7] a stochastic learning automata enables a communication flow control scheme that is used to balance loads in optimistic simulations based on the Time-Warp synchronization algorithm [1]. All these mechanisms are based on the LP migration concept. In other words, LPs are considered the smallest component that can be migrated for simulation load balancing. In general, migrating and re-instantiating LPs between different PEUs may have a significant overhead. Moreover, the adaptation obtained with this approach is coarse grained with respect to the approach obtained by migrating SMEs between LPs. GAIA+ mechanism will adopt the latter approach: we assume the LPs are containers of SMEs, and the load balancing strategy is based on the migration of SMEs between LPs executed over different PEUs.

To the best of our knowledge, our GAIA+ mechanism has the following differences with existing solutions: it improves the computation load balancing and, at the same time, it reduces the communication overheads between LPs by migrating SMEs [3]. In addition, the load balancing mechanism is fully distributed and manages heterogeneous hardware scenarios also in presence of dynamic background load. The latter point represents the main innovation of GAIA+ with respect to preliminary GAIA mechanism introduced in previous works [3,4]. The potential for this innovation is twofold: it enables adoption of COTS architectures by obtaining high performances and by improving the model scalability, and it supports dynamically load balanced simulations without any need for user level configuration.

3 The Adaptive Load Balancing Middleware

The High Level Architecture (HLA) is a general purpose architecture for simulation reuse and interoperability (IEEE Standard 1516) [2]. A HLA-compliant simulation is realized by a set of federates, each federate is a software component that interacts with other federates to form a simulation (federation). Given previous definitions, a federate can be thought of as a LP and viceversa. Many distributed federates can be composed to form simulations, whose interactions are controlled through a distributed middleware called Runtime (RTI). Some implementation criticisms and the lack of basic features as the built-in support for migration-based load-balancing, are the main motivations behind the design and implementation of a new RTI called ARTÌS (Advanced RTI System) [9].

3.1 ARTÌS

The Advanced RTI System (ARTÌS) is a middleware for Parallel and Distributed Simulation (PADS) supporting high degree of model scalability [9]. The design of the middleware is inspired by the IEEE 1516 standard, but new features have been introduced to improve the scalability and the simulation performance. The PADS execution speed is highly influenced by the communication performance: the approach followed by ARTÌS is adaptive and exploits the characteristics of the physical allocation of LPs [5]. ARTÌS supports both the conservative (Chandy-Misra-Briant) and the optimistic (Time Warp) synchronization algorithms. The load balancing mechanism that will be introduced in the following sections is based on a conservative time-stepped synchronization scheme [3]. In [4] it has been shown that the performances of a distributed simulation can be increased by introducing the migration of the simulated entities (SME). A migration based middleware could optimize in adaptive way the simulation execution by reallocating the SME over the LPs. The dynamic reallocation can reduce the communication overhead and moreover can be exploited to improve the computation load balancing. This translates into a reduction of the Wall-Clock Time (WCT) needed to complete the simulation runs. The Generic Adaptive Interaction Architecture (GAIA) is a migration based framework integrated in ARTÌS. The basic task of GAIA is to check the communication pattern of each SME during the simulation execution. A set of heuristics evaluates the communication pattern and triggers the SME reallocation to reduce the communication costs and to improve the load-balancing of the execution architecture. GAIA migrates the highly interacting SMEs within the same LP, by reducing costly inter-LP communication and by increasing the rate of low cost intra-LP communications. The cost of migrating the simulated entities is a key factor to be evaluated in the migration heuristics. An analytic evaluation of this cost is impossible due to the network heterogeneity and the unpredictable behavior of the simulated system.

3.2 GAIA+

The GAIA+ framework is an evolution of the migration mechanism defined in [4]. GAIA+ has been designed and implemented to support the distributed simulation over shared COTS clusters and to enhance the load balancing and communication overheads' reduction in presence of massive models of dynamically interacting SMEs, heterogeneous execution architectures and unpredictable computation and communication (background) loads.

3.3 The Heuristic Migration Policy Definition

The dynamic migration of SME may reduce the message-passing overhead by introducing migration overheads: some analytical or heuristic metrics are required, to be evaluated at runtime, to define "if" and "where" it would be profitable to migrate a SME. The state size of a SME and the amount of "time-locality" of the causal dependencies (LP-local message passing) between correlated SMEs, are the most relevant parameters influencing the migration policy. Specifically, the policy depends on the interaction rate between SMEs, and the overall load balancing policy between the PEUs. By focusing on the network communication-reduction viewpoint, it would be optimal to allocate every SME on a single PEU. Obviously, GAIA+ mechanism has to deal with computation load balancing too, hence the optimal policy would require to dynamically partition in sets the most frequently interacting SMEs, by allocating each set over the available PEUs in load-balanced way. The dynamic load balancing problem is even more complicated by assuming that the CPUs are heterogeneous and subject to unpredictable background load. GAIA+ implements a combination of two low-cost heuristic schemes, that adaptively converge to a balanced solution, under the system assumptions considered in the implementation. The rules for migration heuristic are quite simple and have been improved with respect to the early design of previous work in [3].

3.4 The Heuristic Load-Balancing Policy Definition

The steady state behavior of the migration heuristic in isolation would lead to the asymptotic clustering of all the SMEs over a subset of the available PEUs. This is because the adaptive effect of migrations is focused on the reduction of "external" communication overheads. The migration heuristic must be composed with a computation load balancing heuristic. The load balancing strategy implemented in the previous version of the GAIA framework and defined in [4] was based on some common assumptions: *i*) each CPU executes one single LP, *ii*) all the PEUs are homogeneous and every LP manages the same number of SMEs, *iii*) the execution architecture is dedicated to the simulation and no external background load can interfere with simulation load balancing.

The new version of the framework, GAIA+, has been designed to overcome such limitations. Our previous experience with migration-based distributed simulations shown that a LP overcrowded by many SMEs may be a synchronization bottleneck for the whole simulation. The general load balancing rule that governed the original GAIA migration heuristic allowed only balanced migrations between LPs, in a three-phases migration procedure: in the first phase every LP must claim the number of candidate migrations and their proposed destinations; in the second phase the load balancing condition is evaluated, and in the third phase all the migrations satisfying the load balancing rules are performed. To remove the simplification assumptions described above, it is necessary to introduce some special improvements. Every LP, at each simulation timestep, checks the incoming communication queues to determine which LPs are slow in reaching the synchronization. Some adjacent timesteps are observed to have some confidence on the trends and behaviors of the system, by each LP. The collected data represents a local-LP vision of the foreign-LPs simulation execution. Such information is exchanged by LPs and managed by the distributed GAIA+ middleware components, locally to each LP. GAIA+ middleware infers a global vision and marks the LPs as "slow" or "fast" with respect to the average simulation speed. If the delay between the slow and the fast LPs is significant then the load distribution is not adequate and GAIA+ breaks the adopted general load balancing rule by triggering an *unbalancing exception*, that is, a part of the load has to be migrated from the slow to the fast group of LPs. The *unbalancing exception* allows the LPs marked as slow to migrate a number of SMEs to LPs that are marked as fast, even if this would break the local balancing condition of the involved LPs. However, the implementation of the exception is regulated: the number of SMEs allowed to migrate (referred as *migration set*) is proportional to the difference of speed between slow and fast groups. After defining the size of the migration set, it is necessary to choose the SMEs that will be migrated, and their destinations, in accordance with communication patterns and the migration cost. The implementation of this mechanism has to satisfy a few essential requirements: *i*) it should quickly adapt to heterogeneous hardware with very different CPUs and network performances, *ii*) it should quickly adapt to variations of the background load (both computation and communication), *iii*) it should converge without introducing harmful fluctuations. For the sake of simplicity, in our analysis we assumed that all the SMEs are equivalent in terms of computation-cost per timestep. This assumption is quite common in many simulation models, e.g. in considered wireless ad hoc network models (section 4). If the assumption is not satisfied, the load estimate characterizing each SME has to be taken into account as additional parameter of the heuristic evaluation.

4 Testbed Evaluation

Now we illustrate some key concepts of a testbed model of a wireless system. We assume a high number of Simulated Mobile Hosts (SMHs, that is, the equivalent of general SMEs considered in previous sections), each one following a Random Waypoint (RW) mobility model. This mobility model is far from being real, but this choice was driven by the unpredictable and uncorrelated mobility pattern of SMHs. This is the worst case analysis for the GAIA+ mechanism, because any heuristic definition cannot rely on any assumption about the motion correlation and predictability of SMHs. Space is modeled as a torus-shaped 2-D flat topology, 10.000x10.000 space units, populated by a constant number of SMHs. The torus space topology, indeed unrealistic, is commonly used by modelers to prevent non-uniform SMHs' concentration in any area. The simulated space is without obstacles. The modeled communication between SMHs is a constant flow of ping messages (i.e. constant bit rate), transmitted by every SMH to all neighbors within a wireless communication range of 250 space units.

4.1 Experimental Results

The first set of experiments were executed on a cluster of 3 heterogeneous PEUs: two Dual Xeon Pentium IV 2800 MHz with 3 and 4 GB RAM, respectively, and one Quad Xeon Pentium IV 1500 MHz with 1 GB RAM, connected by a Gigabit Ethernet LAN. We performed multiple runs for each experiment, and the confidence intervals obtained with a 95% confidence level are lower than 5% the average value of the performance index shown. All the performed experiments were initialized with a uniform pseudo-random distribution of 9000 SMHs over a flat topology. The distributed simulation is composed by 3 LPs: each PEU

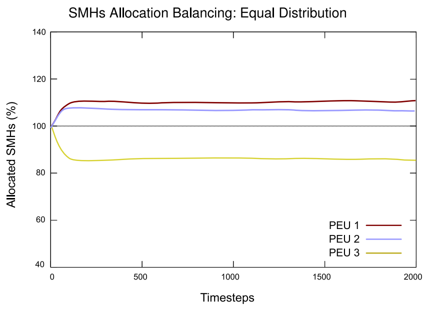

Fig. 1. Initially each PEU simulates the same number of SMHs (3000)

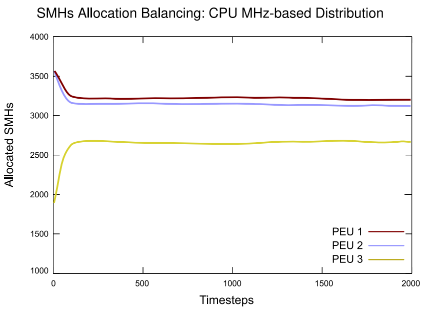

Fig. 2. The initial allocation is based on the CPU MHz of each PEU

manages the execution of one LP. Initially, all the SMHs are randomly and uniformly allocated over the set of PEUs, that is, the model components' allocation is not initialized in a scenario favorable to GAIA+ load-balancing (because PEUs are not homogeneous) and communication-reduction of GAIA+ migration scheme (because modeled wireless hosts with time- and space-correlation are randomly distributed over different PEUs). Testbed evaluation **a**): initially each PEUs allocates the same number of SMHs (3000). All the PEUs are equipped with same generation CPUs (Xeon Pentium IV) but with different clock speed and memory: PEUs 1 and 2 have a 2800 MHz CPU with 3 and 4 GB RAM, respectively, and the PEU 3 has a 1500 MHz CPU with 1 GB RAM. Given the significant difference of speed, we expect that the GAIA+ mechanism will modify the allocation of SMHs for load balancing, reducing the number of SMHs allocated on the slow PEU 3 and increasing the population on the fast PEUs 1 and 2. Figure 1 shows that the expected transient behavior quickly converges to a stable steady state condition, without introducing fluctuations in the number of allocated SMHs. It appears that "fast" PEUs are not homogeneous in performance, as it can be expected due to the difference in the amount of local RAM. The same behavior is confirmed in the results presented in the following.

In the second approach **b**), the initial allocation was based on the nominal performance of the PEUs. In practice, since CPUs are the same generation, it would be possible to roughly allocate the computation load (homogeneous SMEs) proportionally to the clock speed (expressed in MHz). Given this assumption, the 9000 SMEs should be allocated on the fast CPUs (2800 MHz) with 3552 SMHs each, and the slow CPU with 1896 SMHs. In theory, this initial allocation method should be stable under the load balancing viewpoint. Figure 2 demonstrates that this assumption is not confirmed: the GAIA+ framework quickly reacts and reaches a different steady state condition that substantially increases the load on the "slow" PEU. This confirms that load balancing inferred by nominal CPU performance index (like the CPU clock speed) is not adequate.

So far we have considered testbeds with no background load (that is, dedicated cluster). In this case some benchmarks or preliminary simulation tests (like in Figure 1 and 2) could determine a load balanced partition of the SMHs between the execution units involved in the distributed simulation. As discussed in the introduction, this assumption is often unrealistic, as an example with shared clusters with unpredictable background load. In the following we perform some tests of GAIA+ mechanism under the variable background load scenario: the considered system architecture is composed by three Intel Xeon Pentium IV, 2800 MHz, with 2 (PEU1), 4 (PEU2) and 3 (PEU3) GB RAM, respectively. In scenario **c**), we injected a synthetic load over the PEU 1, only. The background load of PEU 1 is shaped as a sinusoidal wave, that is, it is not unpredictable. On the other hand, there is no exploitation of any predictability characteristic in GAIA+: we used this curve because it gradually introduces load variation at different speeds (that is, the derivative of the background load curve shows a variation of slow changes followed by sudden changes). Our analysis goal is to verify the reaction of GAIA+ in presence of slow and fast background load variations. In left-hand figures 3 we show the background load of three PEUs involved in the distributed simulation. The effect on the load balancing mechanism of GAIA+ can be seen on right-hand figures 3: a SMHs re-allocation is realized by GAIA+, by following the shape of the background load in reactive and dynamic way. The right-hand figures report the percentage of allocated SMHs with respect to the initial distribution of 3000 allocated SMH per PEU (100%). SMHs migrated from overloaded PEU1 are fairly re-allocated over PEU2 and PEU3.

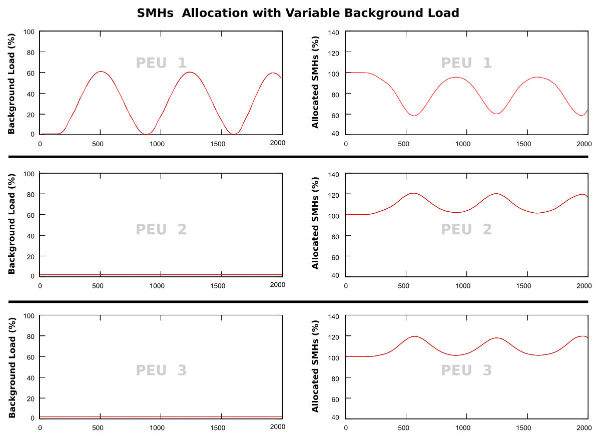

Fig. 3. The effect of the injected background load on the SMHs allocation

These results demonstrate that GAIA+ mechanism can quickly adapt the load partitioning of the SMEs from any initial distributions (e.g. random), and by adaptively reacting to the background load variations. The communication reduction ability obtained by GAIA+ is the same inherited by original GAIA, as shown in [3]. In [3], we also shown that speedup was obtained by the GAIA mechanism with respect to distributed and monolithic simulations. Now we show that GAIA+ mechanism outperforms GAIA under the simulation speedup viewpoint, by reducing the Wall-Clock-Time (WCT) required by simulation runs, under new assumptions that characterize the shared clusters. We analyzed a portion of the execution (1000 steady state timesteps) of the simulation runs of the wireless ad hoc network model, for the three scenarios (a, b and c). Table 1 shows the WCT needed to execute the run portion: we refer to "GAIA+" as a distributed simulation with the new GAIA+ load balancing mechanism enabled and "GAIA" when the old GAIA scheme is used. The results confirm our expectations: the GAIA+ mechanism significantly reduces the WCT in all the analyzed scenarios. The load allocation is quickly balanced over heterogeneous execution scenarios, both in presence of sub-optimal initial allocations (a, b) and in presence of variable background loads (c).

Table 1. WCT (seconds) to complete a simulation run of 1000 timesteps

scenario	GAIA	GAIA+	diff (%)
a	3600	3442	*-4.38%*
b	3983	3568	*-10.41%*
c	5232	4128	*-21.10%*

5 Conclusions and Future Work

In this work we have described GAIA+, a migration-based framework build on top of the ARTÌS middleware. GAIA+ exploits the runtime migration of simulated model entities to concurrently address two main problems of distributed simulation: the reduction of the communication overhead and the load-balancing in the distributed execution architecture. The new GAIA+ framework introduces support for shared and heterogeneous execution architectures possibly characterized by background load. A distributed heterogeneous execution architecture and a wireless ad hoc network model have been used as a testbed for GAIA+ analysis. The performance evaluation has demonstrated that the new heuristics adopted in the GAIA+ middleware can lead to significant reduction in the WCT required to execute the simulation runs.

Future works include new heuristics to address the presence of heterogeneous model entities with different computational requirements, and extended testbed models. Most preliminary Grid-based simulations has revealed poor performances mainly due to the high latency experienced by Internet communications and the lack of control on the nodes. We believe that the ARTÌS and GAIA+ middleware porting on the Grid architecture could possibly contribute to increase the performance of Grid-based simulations.

References

1. Fujimoto, R.M. Parallel and Distributed Simulation Systems. Wiley & Sons, 2000
2. IEEE 1516. Standard for modeling and simulation, High Level Architecture (HLA).
3. Bononi, L., D'Angelo, G., Donatiello, L. HLA-based adaptive distributed simulation of wireless mobile systems. PADS '03: Proceedings of the 17th ACM/IEEE/SCS Workshop on Parallel and Distributed Simulation.
4. Bononi, L., Bracuto, M., D'Angelo, G., Donatiello, L. A New Adaptive Middleware for Parallel and Distributed Simulation of Dynamically Interacting Systems. DS-RT '04: Proceedings of the 8-th IEEE International Symposium on Distributed Simulation and Real Time Applications.
5. Bononi, L., Bracuto, M., D'Angelo, G., Donatiello, L. Analysis of High Performance Communication and Computation Solutions for Parallel and Distributed Simulation. HPCC '05: Springer LNCS Proceedings of the 2005 International Conference on High Performance Computing and Communications.
6. Boukerche, A., Das, S.K. Dynamic Load Balancing Strategies for Conservative Parallel Simulations. PADS '97: Proceedings of the 11th SIGSIM/IEEE/SCS Workshop on Parallel and Distributed Simulation.
7. Choe, M., Tropper, C. On Learning Algorithms and Balancing Loads in Time Warp. PADS '99: Proc. of the 13th Workshop on Parallel and Distributed Simulation.
8. Theodoropoulus, G., Logan, B. An Approach to Interest Management and Dynamic Load Balancing in Distributed Simulation ESIW '01: Proceedings of the 2001 European Simulation Interoperability Workshop.

9. PADS homepage, http://pads.cs.unibo.it
10. Short, J., Bagrodia, R., Kleinrock, L. Mobile wireless network system simulation Wireless Networks 1, 1995
11. Boukerche, A., Tropper, C. A static partitioning and mapping algorithm for conservative parallel simulations. PADS '94: Proc. of the 8th workshop on Parallel and distributed simulation.