Detailed Simulation of Large-Scale Wireless Networks

Michele Bracuto, Gabriele D'Angelo
Dipartimento di Scienze dell'Informazione, Università di Bologna
Mura Anteo Zamboni, 7. 40127, Bologna, Italy

E-mail: {bracuto, gdangelo}@cs.unibo.it

Abstract

In this paper, we present WiFra, a new framework for the detailed simulation of very large-scale wireless networks. WiFra is based on the parallel and distributed simulation approach and provides high scalability in terms of size of simulated networks and number of execution units running the simulation. In order to improve the performance of distributed simulation, additional techniques are proposed. Their aim is to reduce the communication overhead and to maintain a good level of load-balancing. Simulation architectures composed of low-cost Commercial-Off-The-Shelf (COTS) hardware are specifically supported by WiFra. The framework dynamically reconfigures the simulation, taking care of the performance of each part of the execution architecture and dealing with unpredictable fluctuations of the available computation power and communication load on the single execution units. A fine-grained model of the 802.11 DCF protocol has been used for the performance evaluation of the proposed framework. The results demonstrate that the distributed approach is suitable for the detailed simulation of very-large scale wireless networks.

1 Introduction

A simulation is a system that represents or emulates the behavior of another system over time [17]. The simulation technique is of primary importance in the design, implementation and performance evaluation of many real world systems. In designing simulations, one of the key factors is the level of detail of the simulated model. That is the complexity of the representation of real world entities and interactions, within the simulation. The importance of the level of detail is twofold: first of all, the correctness of the simulation results is deeply influenced by the amount of details involved in the representation of the simulated system [22]. In a performance evaluation, an inadequate amount of details in the model representation can lead to misleading or wrong results [16]. On the other side, the level of detail affects the time required for the simulation runs [18]. An increased amount of details in the simulated model translates to many factors: i) more computation is required to evolve the simulation; ii) more memory is necessary to represent the modeled system; iii) an increased amount of communication between the simulated entities. The practical effect is a more complex simulation that requires more time to complete each run.

Many of the systems of interest for research and commercial purposes are large scale, composed of a very large number of entities or parts, and characterized by dynamic nature and evolution. It is expected that this trend will continue in the next years, and consequently will increase the request for highly scalable simulation tools. Wireless networks are often composed of a very large number of nodes and, under the simulation viewpoint, with strict requirements in terms of level of detail [18]. Given the growth rate of wireless technologies, networks composed of hundreds of thousands up to millions of nodes will be widely diffused in the next years. Therefore, we need tools suitable for the detailed simulation of very large scale wireless networks. Many tools used for the simulation of wireless networks have a monolithic design and implementation, that is a single execution unit manages the evolution of the whole simulation [9, 10]. Mainly due to memory constraints and the excessive amount of time required to complete the simulation runs, the approach based on a single execution unit is unable to fulfill the scalability requirements imposed by the simulation of large-scale wireless networks [13]. A lot of memory is required to model a large number of wireless devices, moreover the evolution of this kind of system is characterized by many interactions that have to be generated, delivered and computed. An alternative approach is based on Parallel And Distributed Simulation (PADS) [17], in this case a set of execution units is in charge of the evolution of the simulation. Each execution unit is responsible of a part of the simulation (a subset of the entities that compose the simulated system) and their interactions. To obtain a correct evolution of the distributed simulation, all execution units have to be synchronized during the entire span

1550-6525/07 $25.00 © 2007 IEEE
DOI 10.1109/DS-RT.2007.28

of the simulation. The main advantage of PADS approach is the aggregation of memory and computational resources: an execution architecture composed of many Physical Execution Units (PEU) is able to model very large systems. Furthermore, thanks to the parallel execution of some parts of the model, a speed-up of the simulation can be obtained. The main drawbacks of this approach are: i) the amount of synchronization and communication required for a correct simulation execution and ii) the load-balancing in the PADS architecture. It is obvious that also PADS are deeply affected by the level of detail of simulated models. The availability of many PEUs can reduce the bottleneck due to the computation requirements, but communication in a distributed system is order of magnitude slower with respect to a single execution unit. As said before, in order to obtain correct results the PEUs involved in a distributed simulation have to be synchronized. Increasing the level of detail, means a higher amount of costly communications: to maintain the distributed system synchronized and to deliver the interactions between the simulated entities. In some cases, the communication cost is so high to outweigh the gain obtained by the parallel execution given by multiple PEUs. In this case, the distributed simulation is slower than the monolithic one.

Medium Access Control (MAC) protocols commonly used in wireless networks, require very fine-grained models to represent the state of the shared medium and the behavior of wireless devices [15]. Under the distributed simulation viewpoint, this translates to very frequent synchronizations and a large amount of communication between PEUs. This problem is so critical that many simulation tools are still based on the monolithic approach and therefore unable to simulate very large wireless networks without relying on approximation or aggregation techniques [22]. The main goal of this work is to demonstrate that large scale wireless network (200.000 nodes) can be efficiently simulated following the PADS approach, specifically tailored techniques can improve the communication efficiency and therefore increase the simulation speed. Finally, using appropriate load-balancing schemes it is possible to exploit massively distributed execution architectures composed of Commercial off-the-shelf (COTS) hardware for the simulation of very large-scale wireless networks (1.000.000 nodes). Networked personal computers can be used to build low cost execution architectures that are much more cost-effective than dedicated High Performance Computing (HPC) architectures. The proposed approach permits to share computing facilities with other users (e.g. desktop PCs, university computing labs, etc.) without reserving resources for a single task.

The paper structure is the following: in Section 2 some background concepts and related works about detailed simulation of wireless networks are introduced, in Section 3 we describe the simulator that has been developed, in Section 3.2 are introduced some mechanisms to reduce communication overhead and to improve load-balancing in PADS. In Section 4 is defined the testbed for our performance evaluation, the experimental evaluation is reported and the obtained results are discussed. Finally, Section 5 reports our conclusions and future work.

2 Background and related work

A distributed simulation can be designed as a set of Simulated Model Entities (SME), each SME models a small part of the real system (e.g. a mobile wireless device). Interactions between entities in the real system are modeled as interactions between SMEs. A Logical Process (LP) can be seen as a component of the distributed simulation and the container of one or more SMEs. Each LP is allocated on a Physical Execution Units (PEU) that provides the computational and communication resources. Depending on the hardware performance and the simulator design, the same PEU can allocate one or more LPs. Following this approach the simulator is built as a distributed system: each LP manages the evolution of a part of the simulation. To obtain correct simulation results, all LPs have to be coordinated: the evolution of the simulated model is given by the execution of a synchronization mechanism.

To the best of our knowledge, existing simulation packages have severe limitations on the size and complexity of the wireless networks that can be modeled [21, 24]. The most used simulator for wireless networks, at least in academia, is the Network Simulator 2 (ns-2) [9]. This tools is based on the monolithic approach and therefore is unable to manage very complex models [22]. An enhanced version of ns-2, the Parallel Distributed Network Simulator (PDNS) [6], has been developed to achieve better scalability and performance improvements. PDNS is limited to wired networks, and the traffic simulated at different spatial partitions cannot affect each other [25]. Currently is under development the next major revision of the Network Simulator, called ns-3 [4], one of the most interesting planned features is the support for distributed simulation. In the following, it will be described some tools that are widely used for the simulation of large scale wireless systems. GloMoSim [26] is a simulation environment for wireless and wired network systems based on the Parsec parallel discrete-event simulation kernel [12]. QualNet [7] is a complete and efficient commercial tool derived from GloMoSim. Under the scalability viewpoint, QualNet is reported to scale up to 10's of thousands of nodes [7]. OPNET is an efficient and complete commercial framework for modeling and simulation of wired and wireless communication systems [5]. SWANS is Java-based scalable wireless network simulator built on the top of JiST engine [3]. SWAN [8] is a collection of

models for the simulation of wireless networks that runs on top of the DaSSF [1] kernel. GTNetS is a network simulation environment intended specifically for modeling large-scale topologies [23]. To the best of our knowledge, neither of these tools supports dynamic and adaptive techniques for the reduction of the communication overhead and the dynamic reconfiguration in presence of variable background computational and communication load.

3 Wireless network simulator

In Figure 1 is illustrated the stack-based architecture of the wireless network simulator used in the following performance evaluation (Section 4): on top WiFra models the wireless network (Section 3.3). The middle layer is represented by the GAIA+ framework that provides functionalities to reduce the communication overhead and to adaptively manage load-balancing in distributed simulation (Section 3.2). The core of the simulator is the Advanced RTI System (ARTÌS) middleware (Section 3.1), the main goal of the runtime is to provide an efficient and easy to use environment for parallel and distributed simulation.

Figure 1. Logical architecture of the distributed simulator.

3.1 The Advanced RTI System (ARTÌS)

The Advanced RTI System (ARTÌS) is a middleware for parallel and distributed simulation [11], specifically designed to support high degree of model scalability and execution architectures composed of a large number of PEUs. The design of the middleware has been partially influenced by the High Level Architecture (IEEE 1516) standard [2]. Some new features have been introduced to improve scalability and simulator performance, and a simplified set of Application Programming Interfaces (APIs) has been provided to facilitate the development of PADS. In a distributed simulation, a large part of the interactions are delivered by network communications, therefore the execution speed is highly influenced by the communication performance (e.g. network latency and bandwidth). Consequently, it is of primary importance to consider the characteristics of the physical allocation of LPs and to adaptively adjust the communication behavior with respect to network performance. In example, LPs on symmetric multiprocessing (SMP) or multicore systems should communicate via shared memory. On the other side, LPs connected by LAN, WAN or Internet should rely on the most appropriate communication protocols (e.g. Reliable-UDP, SCTP, TCP, etc.).

Time management is a basic functionality of PADS, offered by the simulation middleware. For the sake of generality, ARTÌS supports both conservative (Chandy-Misra-Bryant) [20] and optimistic (Time Warp) [19] synchronization algorithms. In the detailed simulation of wireless protocols, the natural choice for simulator designers is the conservative time-stepped evolution of the simulated time (Section 3.2.2). Most wireless protocols are based on time slots, for this reason a time-stepped approach facilitates the design and implementation of the simulated model.

3.2 Communication overhead reduction and load-balancing (GAIA+)

Some good ways to speed up distributed simulation would be: i) to have a large number of PEUs, ii) to reduce the synchronization and communication overhead to the bare minimum and iii) to have load-balancing over PEUs. Given the dynamic and unpredictable nature of the distributed simulation environment (e.g. variable network load and latency, presence of background CPU load, etc.) it is not possible to use off-line analytic evaluation to find the best configuration with respect to the requirements described above.

The reduction of the communication overhead and the load-balancing can be seen as different aspects of the same problem and therefore should be addressed concurrently. Under the communication viewpoint, the maximum reduction of the synchronization and communication overhead would be obtained clustering all SMEs in the same PEU. Obviously, this solution is the worst case for load-balancing. In the following of this section we will describe the two main heuristics implemented in the GAIA+ framework, it is worth noting that they do not work in isolation but as part of an integrated mechanism.

3.2.1 Communication overhead

A simulation can be designed as a set of interacting SMEs, the goal of the GAIA+ framework [14] is to enhance the simulation execution by reallocating the SMEs over the available LPs (and therefore PEUs). In this case, the role of the dynamic reallocation is to reduce the communication overhead, and consequently reduce the Wall-Clock Time (WCT) needed to complete the simulation runs. The Generic Adaptive Interaction Architecture (GAIA+) [13, 14] is a migration based framework that exploits the

ARTÌS middleware features. Under the communication front, the main task of GAIA+ is to audit the communication pattern of each SME during the simulation execution. A set of specific heuristics is used to evaluate the communication pattern and to trigger reallocations of SMEs. GAIA+ migrates the highly interacting SMEs to the same LP: clustering together the communication-related SMEs is possible to reduce the costly inter-LP communications and conversely increase the rate of low cost intra-LP communications. In GAIA+, the migration is implemented as the transfer of data structures (i.e. the internal state of simulated entities), the cost of migrating the simulated entities is not negligible and is part of the heuristic. In many practical cases this approach has led to a reduction of the communication costs and a speed-up, both in parallel and distributed simulation [13].

3.2.2 Load-balancing

The migration feature and the presence of many synchronization barriers (each time-step is in fact a synchronization barrier) can be used to introduce a load-balancing mechanism. The proposed mechanism requires very few assumptions from the execution architecture and can correct imbalances that are unrelated to the evolution of the simulation (e.g. unpredictable fluctuations of the available computation power and network load). The design of the GAIA+ load-balancing mechanism is based on a distributed scheme: each LP in the simulation collects the useful information from other LPs, creates a local representation of the distributed system and reacts in presence of imbalances. In detail, a time-stepped simulation can be seen as a series of consecutive synchronization points. Each LP is allowed to move from time-step n to $n + 1$ if and only if all LPs have completed the computation and communication tasks related to time-step n. A major drawback of this synchronization scheme is that the execution speed of the distributed simulation is limited by the speed of the slowest component (that is the "slowest LP problem"). It is worth noting that LP slowness can be due to two main factors: i) the PEU (that allocates the LP) is overloaded; ii) the communication network (used by the LP) has a higher delay with respect to other components of the distributed system. The underlying reasons are very different, but are indistinguishable from outside the LP: the LP is detected as slow. In both cases the reaction should be exactly the same: slow LPs have to migrate some of their locally allocated SMEs to faster LPs. In case i) this action would reduce the computational load; in case ii) the higher delay introduced by the network is balanced by a reduction in local computation. The resulting effect is that the LP will be able to complete the next time-steps in less time, likely reaching the synchronization points with good timing. In extreme cases, very slow LPs (and the related PEUs) will be automatically excluded from the simulator. In this way, GAIA+ will be able to reduce the synchronization overhead due to useless LPs. As said before, the mechanism is totally decentralized, and based on a distributed algorithm that tags the LPs as "slow" and "fast". In accordance with the SMEs migrations triggered by the heuristics that reduces the communication overhead (Section 3.2.1), the load-balancing heuristic triggers additional migrations to load-balance the distributed system. An adequate number of SMEs will be migrated from slow LPs to the group composed of fast LPs. The number of migrations is dynamically defined with respect to the size of the imbalance. The detailed identification of SMEs that are candidates for migrating is quite complex and based on many factors: communication patterns, computational requirements and internal state size.

The proposed mechanism has some interesting characteristics: a) the load-balancing is dynamic and adaptive; b) both computation and communication aspects are considered; c) the mechanism can be tuned: it can be triggered every time-step or delayed, to reduce the related overhead; d) the execution architecture, that is the set of PEUs involved in the distributed simulation, can be very heterogeneous in terms of computational and communication resources. In this case, the load-balancing mechanism will automatically find the adequate level of load for each PEU, depending on its performance; e) the background computation and communication load (e.g. tasks that are unrelated to the simulation execution) can affect the execution architecture but GAIA+ will trigger re-allocations, to improve the balancing; f) the "slowest LP problem" described above is at least partially solved: the execution speed of the distributed simulator is still limited by the slowest LP but the system is now able to correct imbalances due to the evolution of the simulation or caused by external causes. The adequate number of SME to be allocated in each PEU is dynamically determined, depending on runtime conditions and performance of the PEU hardware. A PEU that is overloaded due to tasks that are unrelated to the simulation can be excluded from the simulation, in this way the simulator will not starve waiting for synchronizations. A detailed analysis of the proposed mechanism is out of the scope of this work and has been partially reported in [14].

3.3 Wireless framework (WiFra)

The Wireless Framework (WiFra) is a new library composed of models and auxiliary functions, specifically designed and implemented to simplify the simulation-based performance evaluation of large scale wireless networks. WiFra contains a new detailed model (fine-grained) of the MAC 802.11 DCF protocol that has been used for the performance evaluation described in Section 4. The model

has been implemented in C language for performance reasons and has been designed to comply with the GAIA+ migration-based programming paradigm and the PADS requirements. Details about the testbed configuration and the experimental evaluation will be described in Section 4.1. It is worth noting that, in this case, we are not interested in the evaluation of the specific MAC protocol or other aspects strictly related to the wireless model. The scope of this work is to evaluate performance and scalability of the distributed simulator. In Section 1 we have seen that the level of detail of the simulated model deeply affects the simulator execution speed. We have chosen to implement a fine-grained MAC 802.11 DCF model because it provides a realistic testbed for the simulation of large-scale wireless networks. Some parameters imposed by the protocol are an interesting challenge for distributed simulation. In example, the simulation time-step is imposed by the smallest time parameter defined in the 802.11 DCF protocol that is the Short Interframe Space (SIFS) (10 μs) [15]. Under the distributed simulation viewpoint this translates to extremely frequent synchronizations and therefore a very high communication overhead.

4 Experimental evaluation

In the following, the WiFra simulator will be evaluated in presence of different configurations and execution architectures. First of all, in Section 4.1 will be described the testbed used in the performance evaluation. In Section 4.2 will be shown the scalability results obtained by the simulator while disabling the GAIA+ framework, in this case the WiFra model directly interacts with the ARTÌS middleware. In Section 4.3 the GAIA+ framework will be turned ON and will be evaluated the impact of the communication overhead reduction and load-balancing mechanisms on the simulator. Finally, in Section 4.4 will be demonstrated that following the proposed approach it is possible to build detailed simulations of very large scale wireless networks, using a large number of PEUs.

4.1 Testbed environment

This section will illustrate the main concepts of the testbed model used in the following performance evaluation. We assumed a scenario populated by a high number of mobile wireless devices, referred as Simulated Mobile Host (SMH). In the simulator, each SMH has been implemented as a single SME, therefore under the simulation viewpoint each communication between SMHs will be translated to a set of interactions between SMEs. Each SMHs follows a Random WayPoint mobility model (RWP), this mobility model is far from being real, but it is a good choice to evaluate the GAIA+ mechanism: the uncorrelated mobility pattern of SMHs is not favorable for the clustering mechanisms described in Section 3.2.1. Furthermore, during the initialization phase the SMHs are randomly allocated on the available LPs. Space is modeled as a torus-shaped 2-D flat topology without obstacles, space size depends on the number of SMHs, to have a constant density of SMHs in different tests. On top of the 802.11 DCF protocol we implemented a very simple info-mobility application, the modeled communication between SMHs is a constant flow of fixed size messages (i.e. constant bit rate), transmitted by every SMH to all neighbors within a wireless communication range of 250 meters. Additional details can be found in Table 1. It is worth noting that more complex propagation, mobility and application models would have increased the amount of computation required to complete the simulation runs, without significantly increasing the synchronization overhead. In this case the distributed approach would be highly favored with respect to the monolithic.

Table 1: wireless model parameters

Simulation time-step	10 μs (802.11 SIFS)
MAC layer	IEEE 802.11 DCF
Packet size	1024 bytes
Packet rate	4 pkt/s
Propagation model Transmission range	Free space propagation 250 meters
Simulated area	Variable size fixed density of nodes
Nominal channel bit rate	2 Mbps
Mobility model	Random WayPoint (RWP)
Simulated devices (SMHs)	200.000, 1.000.000

4.2 WiFra scalability

First of all, it has been evaluated the scalability of the simulator without any mechanism to improve the communication and computation load balancing (i.e. GAIA+ framework OFF).

In Figure 2, is shown the scalability of the simulator in a distributed execution environment composed of a variable number of desktop PCs (from 1 up to 8) each one equipped by Dual Core Intel Pentium IV CPU 3.0 GHz with 1 GB of RAM and interconnected by Fast Ethernet network (100 Mbit/s). In reference to the previous definition, each PC is a PEU and, in this case, allocates a single LP. The simulation efficiency was measured in terms of speed-up, that is the rate of execution times between monolithic execution and distributed one. In this scenario, the simulated time was set to 3 seconds and the simulated area was populated by 200.000 SMHs. In Figure 2 is shown a good scalability with $LP = 2$ (i.e. the simulation was executed on 2 different PEUs) with a speed-up value (1.77) that is near the

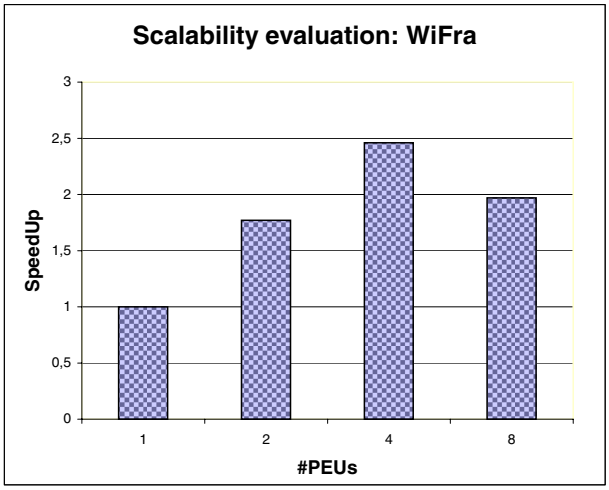

Figure 2. Speed-up obtained by the WiFra framework. LPs = 1, 2, 4, 8. Simulated scenario with 200.000 SMHs.

theoretical maximum (that is 2). With 4 LPs we have a further speed-up increase but the result is quite far from the theoretical maximum (that in this case is 4). Finally, the distributed simulation composed of 8 LPs is still faster than monolithic ($LP = 1$) but shows a speed-up (1.97) lower than $LP = 4$.

These results are in line with our expectations, for many reasons. As described in previous sections, the performance of a distributed simulator can be seen as a trade-off between computation and communication costs. The distributed architecture has a communication cost that should be balanced by the gain of the parallel execution. If the communication cost is lower than the gain we are in presence of a speed-up of the distributed execution ($LP = 2, 4, 8$) with respect to monolithic ($LP = 1$). Conversely, if the communication cost is higher than the gain we have a slowdown. The model used in this performance evaluation is highly populated but dominated by communications (i.e. the synchronization requirements imposed by the 802.11 DCF protocol), in this case each simulated device has very little computational tasks. The total amount of computation required to complete each step of the simulation is sufficient to overload 2 PEUs but insufficient in the case of 4 and 8. In this case, adding more PEUs to the execution architecture increases the communication overhead without any computational gain. A real world scenario characterized by complex user-level applications on top of the wireless protocol would require a higher amount of computation and therefore would benefit of an increased number of PEUs.

A further reason for the speed-up result with $LP = 8$ is due to the "slowest LP problem" described in Section 3.2.2. The cluster of PEUs is homogeneous in terms of hardware (e.g. CPU model and RAM) but with some differences in terms of real performances. Without a load-balancing mechanism (in this case GAIA+ is OFF) a small difference in the background load of each PEU or in the network, leads to a slow-down of the whole simulation. Furthermore, increasing the number of PEUs consequently increases the heterogeneity of the execution architecture.

In accordance with our aim to use COTS computing resources, the performance evaluation was conducted overnight, using a cluster of desktop PCs with the same hardware characteristics but with possible small differences in the installed software and running tasks. The results of many independent runs were collected and carefully scrutinized: we present the mean values.

4.3 The GAIA+ framework

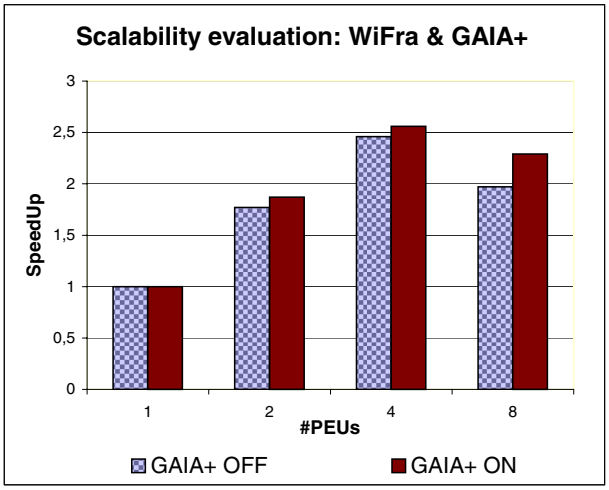

Figure 3. Speed-up obtained by the WiFra framework with GAIA+ OFF / ON. LPs = 1, 2, 4, 8. Simulated scenario with 200.000 SMHs.

The same scenario described in Section 4.2 was simulated placing WiFra on top of the GAIA+ framework. GAIA+ provides a set of features to automatically reduce the communication overhead and improve the load-balancing (as described in Section 3.2). In Figure 3 is shown that with $LP = 1$ (that is a monolithic simulation) GAIA+ is obviously unable to provide speed-up. However, it is interesting that it does not introduce any significant overhead. With $LP = 2$ and $LP = 4$, GAIA+ provides a small increase in the speed-up (up to 5.7%). The small gain in terms of performances shows that the communication in the simulated model is mainly due to synchronizations. As described in Section 3.2.1, GAIA+ migrates the highly interacting SMHs to reduce the communication overhead but is unable to reduce the synchronization overhead

due to the time-stepped synchronization mechanism. In the simulated model, the amount of communication that is related to the model semantic is quite small: the communication ratio of each device is 4 pkt/s with respect to a synchronization time-step of 10 μs (see Table 1). Also in this case, a realistic model with a higher amount of communication related to the model would increase the gain obtained by the GAIA+ framework. With $LP = 8$, GAIA+ has a performance gain of 16.2%, in this case the load-balancing mechanism is able to deal with the heterogenity of the execution architecture. Dynamically adjusting the number of SMHs allocated in each LP, GAIA+ obtains a more uniform system, in terms of execution speed.

4.4 Distributed simulation of very large-scale wireless networks

The last part of this performance evaluation is about the detailed simulation of a very large-scale wireless network, composed of 1.000.000 of nodes. The execution architecture was a cluster of 32 PEUs, Dual Core Intel Pentium IV CPU 3 GHz with 1 GB RAM also in this case interconnected by Fast Ethernet. The distributed simulation was composed by 32 LPs, one for each PEU. Initially each PEU allocated 31.250 SMHs (that is 1.000.000 / 32). During the simulation the load-balancing mechanism has triggered re-allocations to adapt the load of each PEU to the performance of the hardware and background load (e.g. other running simulations, batch tasks of the operating system, etc.).

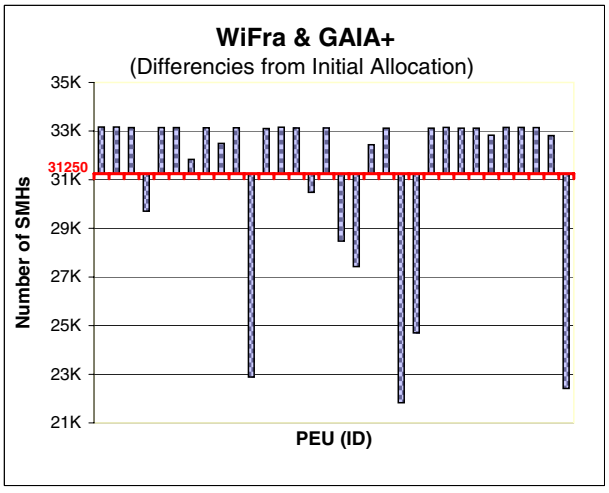

Figure 4. Distributed simulation of 1.000.000 of SMHs (32 PEUs). Allocated SMHs on each PEUs, differences from initial allocation.

In Figure 4 is shown the final allocation of SMHs for each PEU. It worth noting that the effective performance of some PEUs is quite different from the expected performances. PEUs 11, 21, 22 and 32 have a final allocation of less than 25.000 SMHs, with a difference of more than 6.000 units with respect to the initial allocation. Another group of PEUs (4, 15, 17 and 18) have a limited reduction in the number of allocated SMHs. The remaining PEUs have increased the number of SMHs to compensate the reductions in other PEUs. In this last group, the increase is quite uniform across all PEUs.

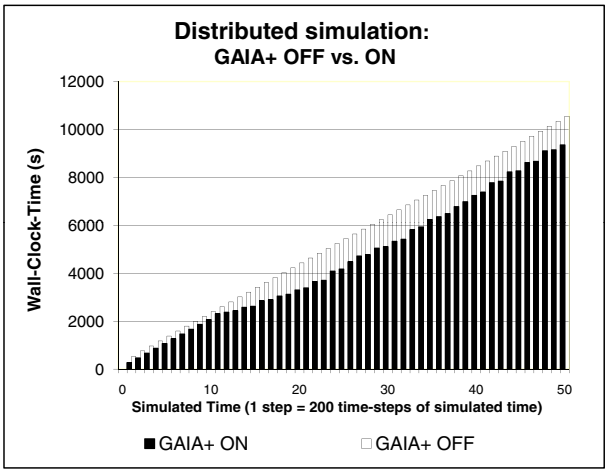

Figure 5. Wall-Clock-Time for the simulation of 1.000.000 of SMHs (over 32 PEUs). Each step in the X-axis equals to 200 time-steps of simulated time.

Under the performance viewpoint, the detailed simulation of wireless networks composed by million of nodes is not feasible following the monolithic approach. A single execution units is unable to represent so large models without severe performance degradation. In Figure 5, we have compared the WCT for the simulation of the scenario described above, with and without the GAIA+ framework (that is GAIA+ ON and OFF). For readability, each step in the graph is obtained by the aggregation of 200 time-steps of simulated time. The results confirm that GAIA+ provides a limited but valuable gain in the time required to complete the simulation. After the warm-up phase, the mechanism is able to speed-up the execution and to provide a gain in terms of performance (up to 21%). The gain is limited but quite stable for the rest of the simulation. For the reasons described in Sections 4.2 and 4.3, this result is very promising. The simulation of more complex and realistic user level-applications would further increase the gain obtained by the GAIA+ framework. Furthermore, execution architectures composed of very heterogeneous PEUs would sharply increase the gain obtained by the GAIA+ framework with respect to a static allocation of SMEs. The warm-up phase

is necessary for the GAIA+ heuristics to collect information about the simulated model and to tune the parameters of the migration-based mechanism. The presence of a very large number of simulated entities is a key issue in the design and implementation of the heuristics. In the considered scenario, the amount of simulated entities is so high that, a badly designed heuristc would be very expensive in terms of computation time and memory requirements.

5 Conclusions and future work

The detailed simulation of large-scale wireless networks is a complex task that involves many aspects and has specific requirements. In this work we have presented WiFra, a new tool for the simulation-based performance evaluation of wireless networks based on the MAC 802.11 DCF protocol. It has been demonstrated that distributed simulation is a feasible approach for the simulation of large-scale networks. Furthermore, an approach based on the migration of the simulated entities can reduce the communication requirements of distributed simulations and concurrently improve the load-balancing in the execution architecture.

Future works include: new models of wireless protocols, the implementation of more realistic user-level applications and protocols to test the performance of the GAIA+ framework in presence of more favorable scenarios, and a careful tuning of the proposed migration-based heuristics to exploit the characteristics of wireless networks. An important planned feature of GAIA+ is the dynamic adaptation of the number of execution units (PEUs) in the distributed simulation architecture. The framework should be able to detect if the execution architecture is overloaded, and therefore activate more PEUs. Conversely, if the execution architecture is underloaded then it should shrink the number of PEUs, to reduce communication and synchronization costs.

References

[1] Dartmouth SSF (DaSSF). http://www.cs.dartmouth.edu/research/DaSSF/.

[2] IEEE Std 1516-2000: IEEE standard for modeling and simulation (M&S) high level architecture (HLA).

[3] JiST / SWANS. http://jist.ece.cornell.edu/.

[4] nsnam: ns3-project. http://www.nsnam.org/.

[5] OPNET Modeler and Wireless Module. http://www.opnet.com/products/modeler/.

[6] Parallel / Distributed ns. http://www.cc.gatech.edu/computing/compass/pdns/.

[7] SNT: QualNet. http://www.qualnet.com.

[8] SWAN: Simulator for Wireless Ad Hoc Networks. http://www.eg.bucknell.edu/swan/.

[9] UCB/LNBL/VINT: the ns-2 network simulator. http://www.isi.edu/nsman/ns/.

[10] OMNeT++: discrete event simulation environment. http://www.omnetpp.org, 2004.

[11] ARTÌS: Advanced RTI System Homepage. http://pads.cs.unibo.it, 2007.

[12] R. Bagrodia and R. Meyer. PARSEC: A parallel simulation environment for complex systems. *IEEE Computer*, 31(10):77–85, 1998.

[13] L. Bononi, M. Bracuto, G. D'Angelo, and L. Donatiello. Performance analysis of a parallel and distributed simulation framework for large scale wireless systems. In *MSWiM '04: Proc. of the 7th ACM Int. Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 52–61. ACM Press, 2004.

[14] L. Bononi, M. Bracuto, G. D'Angelo, and L. Donatiello. An adaptive load balancing middleware for distributed simulation. In *Frontiers of High Performance Computing and Networking - ISPA 2006 Workshops*. Springer, 2006.

[15] L. Bononi, M. Di Felice, M. Bertini, and E. Croci. Parallel and distributed simulation of wireless vehicular ad hoc networks. In *MSWiM '06: Proc. of the 9th ACM Int. Symposium on Modeling Analysis and Simulation of Wireless and Mobile Systems*, pages 28–35, New York, NY, USA, 2006. ACM Press.

[16] D. Cavin, Y. Sasson, and A. Schiper. On the accuracy of manet simulators. In *POMC '02: Proc. of the Second ACM Int. Workshop on Principles of Mobile Computing*, pages 38–43, New York, NY, USA, 2002. ACM Press.

[17] R. Fujimoto. *Parallel and Distributed Simulation Systems*. Wiley & Sons, 2000.

[18] J. Heidemann, N. Bulusu, J. Elson, C. Intanagonwiwat, K. Lan, Y. Xu, W. Ye, D. Estrin, and R. Govindan. Effects of detail in wireless network simulation. In *Proc. of SCS Multiconference on Distributed Simulation*, 2001.

[19] D. Jefferson. Virtual time. *ACM Trans. Program. Lang. Syst.*, 7(3):404–425, 1985.

[20] J. Misra. Distributed discrete event simulation. *ACM Computing Surveys*, 18(1):39–65, 1986.

[21] V. Naoumov and T. Gross. Simulation of large ad hoc networks. In *MSWIM '03: Proc. of the 6th ACM Int. Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems*, pages 50–57, New York, NY, USA, 2003. ACM Press.

[22] L. Perrone, Y. Yuan, and D. Nicol. Simulation of large scale networks ii: modeling and simulation best practices for wireless ad hoc networks. In *WSC '03: Proc. of the 35th Conf. on Winter simulation*, pages 685–693. Winter Simulation Conference, 2003.

[23] G. Riley. Large-scale network simulations with gtnets. In *WSC '03: Proc. of the 2003 Winter Simulation Conference*, 2003.

[24] G. Riley and M. Ammar. Simulating large networks: How big is big enough? In *Proc. of First Int. Conf. on Grand Challenges for Modeling and Simulation*, Jan 2002.

[25] D. Turgut, G. Wang, L. Boloni, and D. Marinescu. Speedup-precision tradeoffs in time-parallel simulation of wireless ad hoc networks. In *DS-RT '06: Proc. of the 10th IEEE International Symposium on Distributed Simulation and Real-Time Applications*, pages 265–268, Washington, DC, USA, 2006. IEEE Computer Society.

[26] X. Zeng, R. Bagrodia, and M. Gerla. GloMoSim: A library for parallel simulation of large-scale wireless networks. In *proc. PADS'98*, 1998.