Distributed Simulation of Large Scale and Detailed Models

Gabriele D'Angelo, Michele Bracuto

Abstract

In this paper, we present a new approach for the distributed simulation of large scale and detailed models. Our approach, based on the migration of simulated entities, enhances the simulator speed jointly addressing two main problems of distributed simulation: the reduction of the communication overhead and the load-balancing in the distributed execution architecture. The proposed method dynamically reconfigures the simulation, taking care of the performance of each part of the execution architecture and dealing with the unpredictable fluctuations of the available computation power and the communication load of each execution unit. In this way, low-cost commercial-off-the-shelf hardware can be used to run fast and cost-effective distributed simulations. A fine-grained model of the 802.11 DCF protocol has been used for the performance evaluation of the proposed approach. The results demonstrate that it is feasible for the detailed simulation of very large scale models such as wireless networks.

Index Terms

simulation; distributed simulation; load balancing; wireless networks.

I. Introduction

A simulation is a system that represents or emulates the behaviour of another system over time [Fujimoto 2000]. Nowadays, the simulation technique is of primary importance in the design, implementation and performance evaluation of many real world systems. The systems of interest are often very large, as number of entities, and characterised by very fine-grained models. In detail, many of them are composed of many entities or parts, and characterised by a very dynamic nature and progress. In designing simulations, one key factor 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 [Perrone et al. 2003]. In a performance evaluation, an inadequate amount of details in the model representation can lead to misleading or wrong results [Cavin et al. 2002]. On the other side, the level of detail affects the time required to run the simulation [Hedidemann et al. 2001]. An increased amount of details translates to many factors:

G. D'Angelo and M. Bracuto are with the Department of Computer Science, University of Bologna, Mura Anteo Zamboni 7, 40127, Bologna, Italy; e-mail addressees {gdangelo,bracuto}@cs.unibo.it.

more computation is required to evolve the simulation, more memory is used to represent the modeled system and an increased amount of communication among the simulated entities is necessary.

The common approach to simulators implementation is based on monolithic design, that is a single execution unit manages the evolution of the whole simulation. In the scenario described above this approach often gives unsatisfactory results. Firstly, a single execution unit is unable to provide the scalability required for the simulation of large-scale models. A lot of memory is necessary to represent the state of a large number of entities, moreover the evolution of such kind of system is characterised by many interactions that have to be generated, delivered and stored. Secondarily, the time required to run such large and complex models is often excessive. Frequently, the amount of time available to obtain the simulation results is very limited, and in some cases faster-than-real-time responses are required [Hybinette et al. 2001].

An alternative approach is based on Parallel And Distributed Simulation (PADS) [Fujimoto 2000], in this case a set of execution units is in charge of the execution of the simulation. Each execution unit is responsible for a part of the simulation (a subset of the entities that compose the simulated system) and their interactions. To obtain a correct execution of the distributed simulation, all the execution units have to be synchronised during the entire span of the simulation. The main advantage of the PADS approach is the aggregation of memory and computational resources: an execution architecture composed of many Physical Execution Units (PEU) is potentially able to model very large systems. Furthermore, thanks to the parallel execution of some parts of the model, a speed-up can be obtained. Two main issues of this approach are: i) the amount of synchronisation and communication required for a correct execution and ii) the computation and communication load-balancing in the PADS architecture.

The cost of communication in distributed systems is orders of magnitude higher than in centralised. To minimise this cost, PADS are often executed on parallel execution architectures (e.g. symmetric multiprocessing, SMP). In fact, the SMP-based communication is characterised by very low latency and high throughput, with respect to LAN or Internet-based distributed systems. On the other hand, SMPs are composed of two or more identical processors connected to a single shared main memory. Therefore, due to their memory constraints, also SMPs are unable to represent very large scale models. Moreover, mainly due to cost reasons, multiprocessors are not widespread.

A more cost-effective solution is based on distributed execution architectures composed of Commercial off-the-shelf (COTS) hardware. For example, networked personal computers can be used to build low cost execution architectures (e.g. desktop PCs, university computing labs, etc.). The performance of this approach is affected by many factors: i) the level of detail of simulated models and in general the amount of computation required for the simulation execution; ii) the cost of the LAN or Internet-based communication; iii) the load-balancing of communication and computation in the distributed architecture. In particular, the execution of detailed models in a distributed architecture requires very frequent synchronisation phases, and this can lead to high communication cost and consequently low execution speed [Hyunok et al. 2007]. To increase the simulator performance, the goal is to reduce at the bare minimum the amount of network based communication and to maintain a good level of load-balancing. In this work we propose a new approach for the detailed (fine-grained) simulation of large scale models. Our approach, that is based on the migration of simulated entities, jointly addresses the two problems described

above: the reduction of the communication overhead and the load-balancing in the distributed architecture. It is worth noting that, in our vision, both aspects are related to the same problem and therefore a combined approach is necessary.

The paper structure is the following: in Section II some background concepts are introduced, in Section III is described the related work about communication cost reduction and load-balancing in parallel and distributed simulation. In Section IV the approach based on entities migration is discussed, and in Section V we consider a real-world case study based on the performance evaluation of a wireless networks simulator. Finally, Section VI reports our conclusions and future work.

II. Background

A distributed simulation can be designed as a set of Simulated Model Entities (SME), in which each SME models a small part of the real system (e.g. a mobile wireless device). The 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 Unit (PEU) that provides the computational and communication resources to the LP (See Figure 1). Depending on the PEU 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 results from the simulation, all LPs have to be carefully coordinated: the evolution of the simulated model depends on the execution of a synchronisation mechanism. Many synchronisation algorithms have been proposed, following different approaches: conservative and optimistic [Misra 1986], [Jefferson 1985]. In the following of our proposal, we assume a simple synchronisation mechanism based on the conservative time-stepped evolution of the simulated time [Fujimoto 2000]. In detail, the distributed simulator does not advance to the next time-step until all simulation activities associated with the current time-step have been completed. Since many communication protocols are based on time slots, the time-stepped approach often facilitates the design and implementation of the related models [Bononi et al. 2008], [Liu et al. 2007].

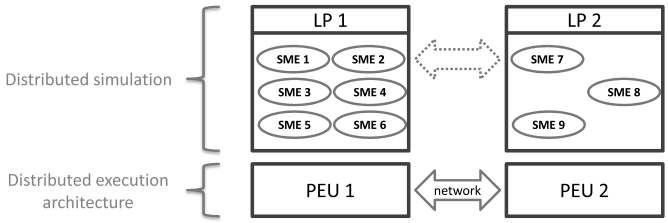

Fig. 1. A distributed simulation composed of 2 LP.

III. Related work

The reduction of the communication cost and the load-balancing are main fields of PADS research. Focusing on the partitioning of simulation models, many different approaches have been proposed to maintain a globalshared-state in distributed simulation, while dynamically filtering the event- and state-information to reduce the communication overhead. Some examples are static partitioning schemes [Fabbri et al. 2000], spheres of influence [Logan et al. 2001], simulation domains [Szymanski et al. 2002], data distribution management [IEEE 1516 Standard 2000], [Raczy et al. 2005], dynamically adaptive partitionions [Kumova 2005] and hierarchical federations [Cai et al. 2001].

Many works have addressed the problem of load-balancing in parallel simulation environments. For example, [Som et al. 2000] propose an approach based on strong groups. Following their definition, a strong group is a collection of model component that strongly affect each other's progress. With respect to our proposal this approach is based on an optimistic synchronisation scheme (i.e. Time Warp, [Jefferson 1985]), implements a much less fine-grained definition of the simulated model and does not directly take care of reducing the communication overhead. An alternative approach is followed by [Deelman et al. 1998]. In this case the two-dimensional simulated space is discretized into a lattice, which is then partitioned and distributed among the different parts of the distributed execution architecture. In [Boukerche et al. 1997], a load balancing scheme for conservative parallel simulations is presented. In this case, the dynamic load balancing algorithm is based upon a process migration mechanism. In [Gan et al. 2000] is presented a load balancing scheme for parallel discrete event simulation that combines both static partitioning and dynamic load balancing. Finally, in [Shanaker et al. 2001] is considered the case of distributed simulation based on conservative approach. In this case, the problem is reduced to an instance of the Task-Allocation Problem (TPA) found in distributed systems. A dynamic reallocation strategy based on LP-reallocation is developed and some experiments based on an Hypercube architecture indicate significant improvements.

Only a few works have examined the problem of simulation model partitioning considering both communication cost and load-balancing requirements in distributed simulation environments. In [Peschlow et al. 2007] is proposed a dynamic partitioning algorithm for optimistic distributed simulation. Our proposal, and the one described in [Peschlow et al. 2007] are both based on the migration of parts of the simulated model, to cope with computation and communication imbalances. Although both approaches have the same goals, the proposed mechanisms are quite different. First of all, the mechanisms are aimed for diverse synchronisation algorithms. Furthermore, they are both based on a migration-based approach but coping with different granularity of the migrated objects (LP in the case of [Peschlow et al. 2007] and single simulated entities in our case). Most importantly, the mechanism proposed in [Peschlow et al. 2007] is based on the calculation, in each host, of performance estimates for both local computation and communication workload. The measurements are collected by a global dynamic partitioning instance (DynPart) that reacts to imbalances instructing migrations. Instead, we propose a distributed mechanism that uses the information provided by the synchronisation algorithm to detect and react to the imbalances.

IV. Adaptive distributed simulation

A fast distributed simulator should be able to use a large number of PEUs, to reduce the synchronisation and communication cost to the bare minimum and to load-balance the cluster of 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 of the simulation with respect to the above described requirements. For example, the partitioning of the simulation model deeply affects the computational and communication load of each PEU. Moreover, the heterogeneity of PEUs and the dynamic load of the execution environment further complicate the search for a good configuration. Given that, in this case, every form of static partitioning is inadequate: a dynamic and adaptive approach is required.

We aim to design a tool that automatically reconfigures the distributed simulator, adapting the model partitioning and the execution architecture to the runtime requirements. In our opinion, the details about entities allocation and execution platform should be totally transparent to the simulator's user. Given that, the starting point is the design and implementation of mechanisms for the reduction of the communication overhead and the load-balancing management in the distributed architecture. It is worth noting that both can be seen as different aspects of the same problem and therefore a joint approach is necessary. For example, under the communication viewpoint, the maximum reduction of the synchronisation and communication overhead would be obtained clustering all SMEs in the same PEU. Obviously, this solution is the worst case for load-balancing.

The proposed approach, described in the following of this section, can be seen as based on two parts with different goals: i) the communication overhead-reduction mechanism, and ii) the load-balancing mechanism, both implemented in the distributed simulation middleware and supervising the simulator execution.

A. Communication overhead reduction

Following the approach based on simulated entities described in Section II, a simulation can be seen as a set of interacting SMEs. Each entity will model the evolution of a very small part of the system and will interact with other entities using messages. In detail, SMEs that are allocated on the same LP will be able to interact via low latency and low overhead networks (e.g. shared memory), that is intra-LP communication. Conversely, SMEs that are allocated on different LPs (i.e. extra-LP communication) will experience the cost of LAN-based communications. To reduce the communication cost and therefore enhance the simulation execution, we propose to adaptively reallocate the SMEs over the available LPs (and therefore PEUs). Our proposal is to audit communication pattern of each SME during the simulation execution and to evaluate if reallocations of SMEs are necessary. In this way, the highly interacting SMEs can be migrated to the same LP. Clustering together the communication-related SMEs is possible to reduce the costly inter-LP communication and conversely increase the rate of low cost intra-LP communication. In this case, the role of the dynamic reallocation is to reduce the communication overhead. In detail, the migration can be 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 has to be considered. As said before, this mechanism is composed of two main heuristics: i) the "base heuristic" and ii) the "group heuristic".

1) Base and group heuristics: The base heuristic analyses the communication pattern of each SME and determines if it should be migrated. At the end of each simulation time-step, during the synchronisation phase, it is analysed the communication pattern of all SMEs that have sent at least one interaction. A SME that has much of its interactions delivered to SMEs that are outside its "local" LP (that is extra-LP communication) is a good candidate for migration. The destination of the migration is the LP that amounts of the larger percentage of outbound interactions. In detail, let a tagged $SME(j)$ be executed on the i-th LP. Let us define $R_{j\_e} = M_e/M_i$ as the ratio of the M_e "external" interactions sent to the e-th LP, with respect to the number (M_i) of "local" interactions sent within the local (i-th) LP. The resulting ratio R_{j_e} is evaluated for every foreign e-th LP. If the maximum ratio obtained is greater than a global threshold-value K ("migration factor", default value $K = 3$), then the corresponding LP is chosen as the candidate destination for the $SME(j)$ migration in the next time-step. No migration is performed, otherwise [Bononi et al. 2004b]. This scheme is integrated in a sliding-windows mechanism that introduces a weighting method. The weight of each interaction depends on its aging and payload size. To obtain good results, the heuristic needs to have up-to-date data as input. Interactions that are too old should not be included in the data set, since they are related to a simulated model state that could be very different from the current one. At the same time, focusing only on the interactions delivered in the current state, drastically reduces the possibility to perform trend analysis and could lead to underreaction or overreaction in the adaptive mechanism. For example, in the simulation of mobile wireless networks, mainly due to mobility factors in the simulated model, interactions (e.g. the events modeling the wireless transmissions) older than a specific threshold have to be discarded or underweighted. A simple way, to implement the weighting mechanism, is to discard all interactions older than a threshold-value IH (called "interaction-horizon", default value $IH = 3$ sec. of simulated time) and to weight the interactions in inverse proportion to their distance from current simulated time. In presence of interactions with different payload size, it is applied a correction factor that is proportional to the size, with respect to the average of interactions. Tuning parameters and migration timeouts are introduced to stabilise the system. For example, a fast moving SME would change very frequently its interaction pattern and therefore introduce a high number of worthless migrations. The migration factor (K) introduced above, can be modified by the simulation modeller, and has the role to tune the amount of migrations in the system. In some sense, it controls the adaptivity of the distributed simulation. Furthermore, a migration timeout has been introduced to control the number and the timing of migrations of each SME. After each migration, the timer of the migrated SME is set to a predefined value (default = 10), each interaction that is sent by the SME has the effect to decrement the counter, and a new migration can be performed only when the counter gets to zero.

The results of the basic heuristic are analysed and modified by the group heuristic. In this case, the goal is to find and evaluate the groups of SMEs that are candidate for migration. Single SMEs that are not candidate could be migrated due to the migration of all SMEs that are interacting with them. In these terms, the group heuristic tries to achieve a higher level of abstraction with respect to the base one. Let us define "interaction group" a set of SMEs that are interacting. For example, some wireless devices in the same location. The implementation of the group heuristic, first of all, tries to determine the interaction groups in the migrating SMEs. For each interaction group

that is found, the heuristic analyses the communication pattern of each SME that is in the group. It determines if other SMEs (that are local to the LP) should be migrated, with the aim to maintain the clustering of the whole group in the same LP.

The approach introduced above, which has been partially described in [Bononi et al. 2003], [Bononi et al. 2004b], in many practical cases has led to a significant reduction of the communication costs and a speed-up, both in parallel and distributed simulation [Bononi et al. 2004a]. It is worth noting that, the implemented heuristics could be further refined, and the default threshold values could be tuned case-by-case (depending mainly on the simulated model characteristics) or a further auto-adaptation mechanism could be added. Conversely, our goal is to demonstrate that very simple and quite generic heuristics, based on reasonable tuning parameters, are sufficient to enhance the distributed simulation execution.

B. Load-balancing

The load-balancing mechanism is based on the migration of simulated entities and the presence of synchronisation barriers during the simulation execution. In presence of a time-stepped synchronisation mechanism, each time-step is, in fact, a synchronisation barrier. On the other hand, if synchronisation is based on the Chandy-Misra-Bryant [Misra 1986] algorithm, specific synchronisation points can be introduced. The proposed mechanism requires very few assumptions from the execution architecture and can also correct imbalances that are unrelated to the evolution of the simulation (e.g. unpredictable fluctuations of the available computation power and network load).

1) Mechanism design and implementation: As said before, the proposed mechanism relies on the instrumentation of the synchronisation algorithm implemented in the distributed simulation middleware. It is automatically triggered at every predetermined time interval (the default value is 50 simulated time-steps), this interval can be tuned depending on the characteristic of the simulated model and the execution architecture. Focusing on the synchronisation, a time-stepped simulation can be seen as a series of consecutive synchronisation points, in which 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 synchronisation scheme is that the execution speed of the distributed simulation is limited by the speed of the slowest component (i.e. “slowest LP problem”). It is worth noting that LP slowness may 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 synchronisation points with good timing. In extreme cases, very slow LPs are automatically excluded from the simulator. In this way, the proposed mechanism can reduce the synchronisation cost due to useless LPs.

More in detail, the implementation of the load-balancing mechanism is based on a distributed scheme. Each LP

in the simulation: i) collects the useful data from the synchronisation algorithm (i.e. the execution speed of other LPs), ii) creates a local representation of the distributed system and iii) reacts in presence of imbalances. In phase i), each LP compares its arrival to the synchronisation barriers with respect to all other LPs. Using this information, in phase ii), each LP calculates a sort of rating that is based on the execution speed of the LPs. The LPs that are on top (default: top 20% of the list) are tagged as "fast", conversely the bottom LPs (default: bottom 10%) are tagged as "slow". In phase iii), if the LP is marked as "slow" then it is enabled to migrate some local SMEs to the LPs marked as "fast". In practice, the load-balancing mechanism triggers additional migrations to improve the balancing of the distributed system. Each LP tagged as "slow" has to determine: a) how many SMEs to migrate, b) the LP or the LPs that should receive the migrating SMEs, and c) which are the SMEs that should be migrated. The number of migrations, point a), is dynamically defined with respect to the size of the imbalance. Each LP calculates the Wall-Clock-Time (WCT) required to complete the simulation of a single time-step (Local Execution Time, LET). Again using the data obtained from the synchronisation algorithm, the LP calculates the Mean Execution Time (MET) that is the mean WCT required from other LPs to simulate a single time-step. Assuming that all the SMEs allocated in a LP are homogeneous in terms of computation load, the number of SMEs to migrate is determined as proportional to the size of the imbalance (i.e. $LET - MET$), and with respect to the total amount of SMEs that are allocated in the LP. In presence of SMEs with heterogeneous computational load, the quantification of the SMEs to migrate is more complex but still possible. In this case, the mechanism has to audit, step-by-step, the execution of the LP and needs to determine the amount of CPU used by each SME that is locally allocated. With this information available, the problem reduces to a variant of the well-known 0-1 knapsack problem that is solved using dynamic programming. Focusing on point b), many approaches can be used to choose the LPs that should accommodate the migrating SMEs. Choosing, as recipient, a random LP in the "fast group" could introduce severe fluctuations in the distributed system, particularly in execution architectures composed of few LPs. Our choice is to uniformly distribute the migrating SMEs among all LPs in the group. Finally, the identification of which SMEs should be migrated, point c), is done in accordance with the SMEs migrations triggered by the mechanism that reduces the communication overhead (Section IV-A1) and again with the goal of clustering in the same LP the highly interacting simulated entities. It is worth noting that, in many simulated models, a bad choice of migration candidates could negatively affect the communication overhead. Again, this evaluation is done inspecting the communication pattern of SMEs but, in this case, other factors (e.g. computational requirements and internal state size of each SME), have to be considered.

2) Properties of the mechanism: The described mechanism has some interesting characteristics: a) the load-balancing is dynamic and adaptive; b) computation and communication aspects are both considered; c) the mechanism can be tuned: it can be triggered every time-step or delayed, to control and reduce the introduced 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 runtime performance; e) the background computation and communication load (e.g. tasks that are unrelated to the simulation execution) can interfere with the simulation

execution, but the load-balancing mechanism will trigger re-allocations to improve the partitioning; f) the "slowest LP problem" described above is partially solved: the execution speed of the distributed simulator is still limited by the slowest LP, but now the system is able to correct imbalances due to internal or external factors. 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, so that the simulator will not starve waiting for synchronisations. Some details about a preliminary version of the proposed mechanism can be found in [Bononi et al. 2006]. It is worth noting that the heuristics composing the adaptive mechanism have to be rather simple and very efficiently implemented. Otherwise, in presence of models composed of a large number of simulated entities (see Section V-D), the computational cost of the heuristics would slow down the simulator.

V. Case study: wireless networks

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 [Hedidemann et al. 2001]. 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.

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 behaviour of wireless devices. Under the distributed simulation viewpoint, this translates to very frequent synchronisations and a large amount of communication between PEUs [Hyunok et al. 2007]. 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 [Perrone et al. 2003], [Guo et al. 2000].

The main goal of this case study is to demonstrate that large scale wireless network (200.000 nodes) can be efficiently simulated following the PADS approach when specifically tailored techniques are used to improve the communication efficiency. Moreover, implementing 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 having to reserve the computational resources for a single task.

Figure 2 illustrates the stack-based architecture of the wireless network simulator used in the following performance evaluation (Section V-D): on top WiFra models the wireless network (Section V-C). 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 V-B). The core of the simulator is the Advanced RTI System (ARTÌS) middleware (Section V-A), whose main goal is to provide an efficient and easy to use environment for parallel and distributed simulation.

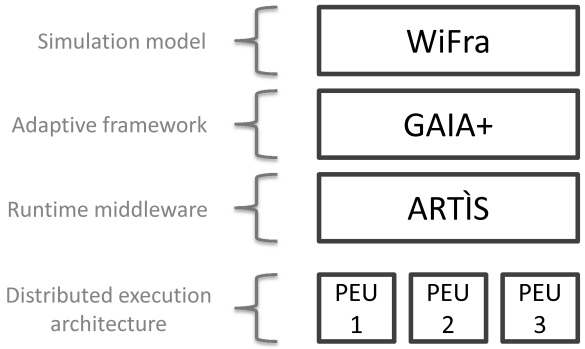

Fig. 2. Logical architecture of the distributed simulator.

A. The Advanced RTI System (ARTÌS)

The Advanced RTI System (ARTÌS) is a middleware for parallel and distributed simulation [ARTÌS Homepage 2008], specifically designed to support high degree of model scalability and execution architectures composed of a large number of PEUs. The design of the middleware is partially influenced by the High Level Architecture [IEEE 1516 Standard 2000]. The implementation has been used as a testbed for the design and development of new features such as: simulation cloning and concurrent replication, communication marshalling, and the study of specifically tailored data structures for the management of simulation events. 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 behaviour with respect to network performance. For example, LPs on symmetric multiprocessing (SMP) or multi-core 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.). As seen before, synchronisation is a main issue in PADS, therefore the simulation middleware has to provide time management services to the simulation components composing the distributed simulation. For the sake of generality, ARTÌS supports both conservative (Chandy-Misra-Bryant, time-stepped) [Misra 1986], [Fujimoto 2000] and optimistic (Time Warp) [Jefferson 1985] synchronisation algorithms. The middleware is freely available for research purposes and can be downloaded from the [ARTÌS Homepage 2008].

B. Generic Adaptive Interaction Architecture (GAIA+)

The Generic Adaptive Interaction Architecture (GAIA+) [Bononi et al. 2003], [Bononi et al. 2004a], [Bononi et al. 2006] is a migration based framework that uses the services provided by the ARTÌS middleware. GAIA+ implements the overhead reduction and load-balancing mechanisms described in Sections IV-A and IV-B. The framework provides

to the simulation developer an easy-to-use entity based paradigm for the definition of models. Following this approach, the level of abstraction provided to the developer is relatively high. In this sense, the ARTÌS middleware is completely transparent to the developer. GAIA+ provides to the simulation model the communication services and the support for entity migration. In this case too, the details about load-balancing and communication cost reduction are transparent. Some parameters for the tuning of advanced and specific features can be tuned at runtime by the model developer (e.g. the migration timeout described in Section IV-A). The GAIA+ framework has been used for the implementation of many simulation models such as: wireless ad hoc networks, sensor networks, scale-free networks and agent-based cooperative multi-agent systems. In the following section, it will be described the implementation of a detailed wireless model based on the services provided by the framework.

C. 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 provides a detailed model (fine-grained) of the MAC 802.11 DCF protocol that has been used for the performance evaluation described in Section V-D. The model has been implemented in C language for performance reasons and has been designed to comply with the GAIA+ migration-based programming paradigm. Details about the testbed configuration and the experimental evaluation will be described in Section V-D1. 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 case study is to evaluate the performance and scalability of the proposed approach in a real-world setting. In Section I we have seen that the level of detail of the simulated model deeply affects the execution speed. For this reason, a fine-grained MAC 802.11 DCF model has been implemented, providing a realistic testbed for the simulation of large-scale wireless networks. Some parameters imposed by the protocol are an interesting challenge for distributed simulation. For 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). Under the distributed simulation viewpoint this translates to extremely frequent synchronisations and therefore a very high communication overhead [Hyunok et al. 2007].

D. Experimental evaluation

In the following, the WiFra simulator will be evaluated in presence of different configurations and execution architectures. First of all, in Section V-D1 will be described the testbed used in the performance evaluation. In Section V-D2 will be shown the scalability results obtained by the simulator while disabling the GAIA+ framework (i.e. disabling the overhead reduction and load balancing mechanisms). In Section V-D3 the GAIA+ framework will be turned ON and the impact of the proposed mechanisms on the simulator performance will be evaluated. Finally, in Section V-D4 we will demonstrate that following the proposed approach it is possible to build detailed simulations of very large scale wireless networks, using a large cluster of PEUs.

1) Testbed environment: This section illustrates the main aspects 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 Hosts (SMHs). 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) [Bettstetter et al. 2002], 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 favourable for the clustering mechanisms described in Section IV-A. Furthermore, during the initialisation 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 neighbours 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 synchronisation overhead. In this case the distributed approach would be highly favoured 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

2) WiFra scalability: First of all, it has been evaluated the scalability of the distributed simulator in the standard configuration, that is without any mechanism to improve the communication and the computation load balancing (i.e. GAIA+ OFF).

In Figure 3, it 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

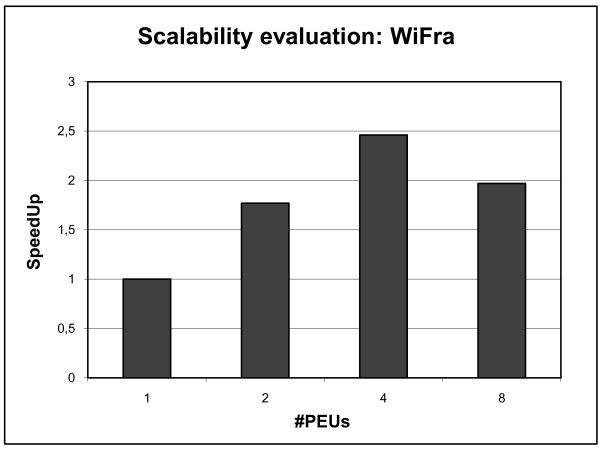

Fig. 3. Speed-up obtained by the WiFra framework. LPs = 1, 2, 4, 8. Simulated scenario with 200.000 SMHs.

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 3 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 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 with $LP = 4$.

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 slow-down. The model used in this performance evaluation is highly populated but dominated by communications (i.e. the synchronisation 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 characterised by complex user-level applications on top of the wireless protocol would require a higher amount of computation and therefore would benefit of a larger number of PEUs.

A further reason for the speed-up result with $LP = 8$ is due to the “slowest LP problem” described in Section IV-B. 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 scrutinised: we present the mean values.

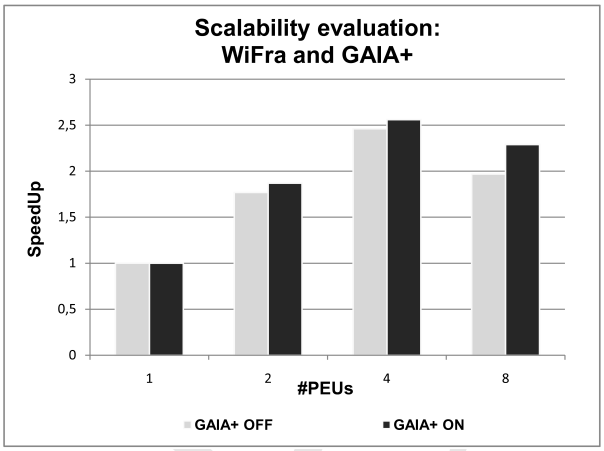

Fig. 4. Speed-up obtained by the WiFra framework with GAIA+ OFF / ON. LPs = 1, 2, 4, 8. Simulated scenario with 200.000 SMHs.

3) Performance of the GAIA+ framework: The same scenario described in Section V-D2 was simulated enabling the GAIA+ framework. In Figure 4 is shown that with $LP = 1$ (that is a monolithic simulation) GAIA+ is obviously unable to provide a 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 synchronisations. As described in Section IV-A, GAIA+ clusters the highly interacting SMHs to reduce the communication overhead but is unable to reduce the synchronisation overhead due to the synchronisation 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 synchronisation 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 reacts to the heterogeneity 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) 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 of 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 users' tasks, batch tasks of the operating system, etc.).

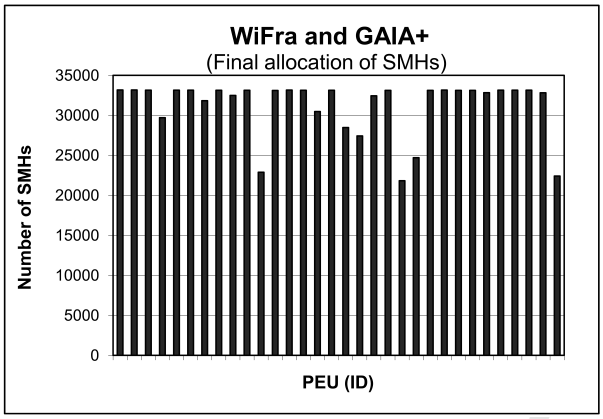

Fig. 5. Distributed simulation of 1.000.000 of SMHs (32 PEUs). Allocated SMHs on each PEUs, final allocation.

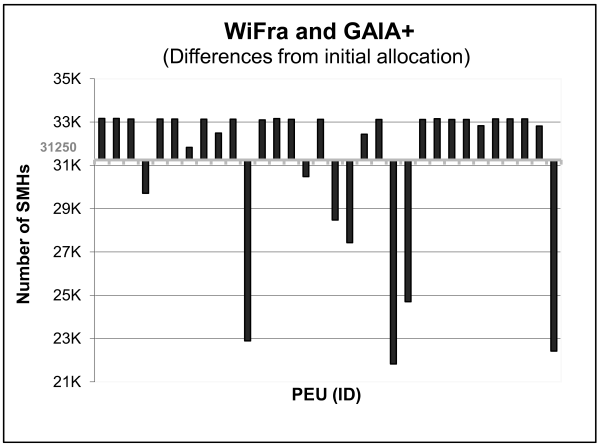

Fig. 6. Distributed simulation of 1.000.000 of SMHs (32 PEUs). Allocated SMHs on each PEUs, differences from initial allocation.

In Figure 5 is shown the final allocation of SMHs for each PEU, the difference from the initial allocation is relevant (see Figure 6). It is 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.

Under the performance viewpoint, the detailed simulation of wireless networks composed of million of nodes is not feasible following the monolithic approach. A single execution units is unable to represent so large models

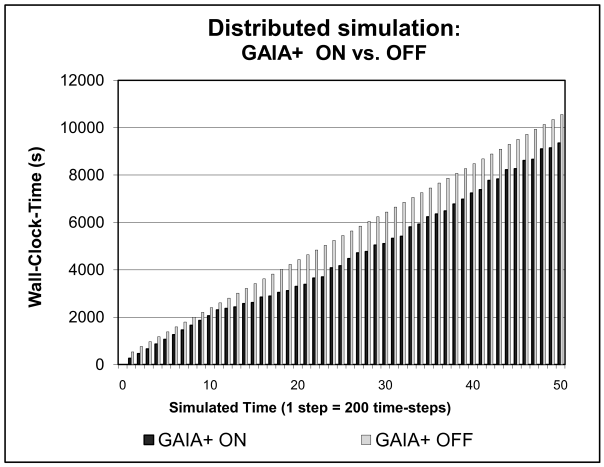

Fig. 7. Wall-Clock-Time for the simulation of 1.000.000 of SMHs (over 32 PEUs). Each step in the X-axis equals to 200 time-steps of simulated time.

without severe performance degradation. In Figure 7, 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 V-D2 and V-D3, 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. As described in Section IV-B, 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 heuristic would be very expensive in terms of computation time and memory requirements.

5) Evaluation of the proposed mechanisms: As stated in Section I, in our vision, the reduction of the communication overhead and the load-balancing are different aspects of the same problem. Therefore our proposed solution is a combined approach that is based on two main mechanisms (i.e. communication overhead reduction and load-balancing management), each one implemented using a set of heuristics. In previous sections, we have shown the experimental evaluation of our proposal in presence of different scenarios and execution environments. The estimate of the contribution of each mechanism to the performance results is a rather complex task due to the many variables and the influencing factors that should be taken in account to obtain an unbiased evaluation.

The performance of the communication overhead reduction mechanism (Section IV-A) is highly influenced by

the characteristics of the simulated model. For example, the amount of communication required by the simulation model with respect to computation. In presence of a CPU-intensive model, the gain obtained by the reduction of the communication overhead is negligible with respect to the computation time. On the other hand, it is worth noting that this part of GAIA+ can reduce the communication cost of the interaction among simulated entities but is unable to reduce the communication cost due to the synchronisation of the distributed architecture. Therefore, in presence of simulation models with very frequent synchronisation barriers, the positive effect of the mechanism could be highly reduced. For these reasons, the wireless model introduced in Section V-D1 results as highly unfavourable for the evaluation of the communication overhead reduction mechanism. In previous works [Bononi et al. 2004a], [Bononi et al. 2004b], we demonstrated that, in presence of less extreme scenarios, the proposed approach leads to significant speed-up of parallel and distributed simulations. Finally, in presence of imbalances in the hosts running the simulation, the load-balancing mechanism (Section IV-B) is the main responsible of the speed-up obtained by GAIA+. This is due to the characteristics of the conservative synchronisation algorithm used in the simulation. The presence of synchronisation barriers requires that each LP has to reach the barrier. Therefore, even a single LP that is slow in reaching the barriers is a huge bottleneck for all others that are idle while waiting for synchronisation.

VI. Conclusions and future work

In this work we have proposed a new approach, based on the migration of simulated entities, to increase the performance of distributed simulations. In our vision, two major problems of distributed simulation, the communication overhead reduction and the load-balancing, have to be seen as different aspects of the same problem. Consequently, a joint approach is strictly necessary.

Our solution, based on specifically tailored heuristics, reallocates the simulated entities clustering the highly interacting components of the simulation and reacting to computation and communication imbalances. A fine-grained model of a wireless protocol has been used for the performance evaluation of the proposed approach. The results demonstrate that distributed simulation is a feasible approach for the simulation of large scale models. Furthermore, big clusters of desktop PCs can be used for the detailed simulation of very large-scale wireless networks, composed of 1.000.000 of nodes.

As a future work we plan to further increase the adaptivity of the proposed mechanism: the distributed simulation architecture should dynamically adapt the number of execution units with respect to the runtime requirements. For example, 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 synchronisation costs. We envision a mechanism that automatically reconfigures the simulation, monolithic or distributed and vice versa, depending on the runtime requirements and the dynamic characteristics of the execution architecture. Following this approach, the usability of simulation tools would be greatly improved, fostering a wider adoption of the distributed simulation techniques.

References

[ARTÌS Homepage 2008] ARTÌS: Advanced RTI System Homepage (2008) URL: http://pads.cs.unibo.it.

[Bettstetter et al. 2002] Bettstetter, C., Hartenstein, H. and Perez-Costa, X. (2002) 'Stochastic properties of the random waypoint mobility model: epoch length, direction distribution, and cell change rate', *Proc. 5th ACM Int. Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, ACM Press, Atlante, Georgia, pp.7–14.

[Bononi et al. 2008] Bononi, L., Di Felice, M., D'Angelo, G., Bracuto, M. and Donatiello, L. (2008) 'MoVES: A framework for parallel and distributed simulation of wireless vehicular ad hoc networks', *Elsevier's Computer Networks (ComNet) Journal*, Vol. 52, No. 1, pp.155–179.

[Bononi et al. 2006] Bononi, L., Bracuto, M., D'Angelo, G. and Donatiello, L. (2006) 'An Adaptive Load Balancing Middleware for Distributed Simulation', *Proc. ISPA 2006 Workshops*, Frontiers of High Performance Computing and Networking, Sorrento, Italy.

[Bononi et al. 2004b] Bononi, L., Bracuto, M., D'Angelo, G. and Donatiello, L. (2004) 'A New Adaptive Middleware for Parallel and Distributed Simulation of Dynamically Interacting Systems', *Proc. 8th IEEE Int. Symposium on Distributed Simulation and Real Time Applications*, IEEE Press, Budapest, Hungary.

[Bononi et al. 2004a] Bononi, L., Bracuto, M., D'Angelo, G. and Donatiello, L. (2004) 'Performance analysis of a parallel and distributed simulation framework for large scale wireless systems', *Proc. 7th ACM Int. Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, ACM Press, Venice, Italy, pp.52–61.

[Bononi et al. 2003] Bononi, L., D'Angelo, G. and Donatiello, L. (2003) 'HLA-based adaptive distributed simulation of wireless mobile systems', *Proc. 17th ACM/IEEE/SCS Workshop on Parallel and Distributed Simulation*, IEEE Press, San Diego, California.

[Boukerche et al. 1997] Boukerche, A. and Das, S.K. (1997) 'Dynamic load balancing strategies for conservative parallel simulations', *ACM SIGSIM Simulation Digest*, Vol. 27, No. 1, pp.20–28.

[Cai et al. 2001] Cai, W., Turner, J.S. and Gan, B.P. (2001) 'Hierarchical federations: an architecture for information hiding', *Proc. 15th ACM/IEEE/SCS Workshop on Parallel and Distributed Simulation*, IEEE Press, Lake Arrowhead, California, pp.67–74.

[Cavin et al. 2002] Cavin, D., Sasson, Y. and Schiper, A. (2002) 'On the accuracy of MANET simulators', *Proc. 2th Int'l Workshop on Principles of Mobile Computing*, ACM Press, Toulouse, France, pp.38–43.

[Deelman et al. 1998] Deelman, E. and Szymanski, B.K. (1998) 'Dynamic load balancing in parallel discrete event simulation for spatially explicit problems', *ACM SIGSIM Simulation Digest*, Vol. 28, No. 1, pp.46–53.

[Fabbri et al. 2000] Fabbri, A. and Boukerche, A. (2000 'Partitioning parallel simulation of wireless networks', *Proc. Winter Simulation Conference*, IEEE Press, Orlando, Florida, pp.1449–1457.

[Fujimoto 2000] Fujimoto, R.M. (2000) 'Parallel and Distributed Simulation Systems', *Wiley & Sons*.

[Gan et al. 2000] Gan, B.P., Low, Y.H., Jain, S., Turner, S.J., Cai, W., Hsu, W.J. and Huang, S.Y. (2000) 'Load balancing for conservative simulation on shared memory multiprocessor systems', *Proc. fourteenth workshop on Parallel And Distributed Simulation*, Bologna, Italy, pp.139–146.

[Guo et al. 2000] Guo, Y., Gong, W. and Towsley, D. (2000) 'Time-stepped hybrid simulation (TSHS) for large scale networks', *Proc. IEEE Infocomm 2000*, Tel Aviv, Israel, vol.2, pp.441–450.

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

[Hybinette et al. 2001] Hybinette, M. and Fujimoto, R.M. (2001) 'Cloning parallel simulations', *ACM Trans. Model. Comput. Simul.*, Vol. 11, No. 4, pp.378–407.

[Hyunok et al. 2007] Hyunok, L., Manshadi, V., Cox, D.C. and Cheung, N.K. (2007) 'High Fidelity Simulation of Mobile Cellular Systems with Integrated Resource Allocation and Adaptive Antennas', *Proc. IEEE Wireless Communications and Networking Conference*, Hong Kong, China, pp.3210–3215.

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

[Jefferson 1985] Jefferson, D.R. (1985) 'Virtual time', *ACM Transactions Program. Lang. Syst.*, Vol. 7, No. 3, pp.404–425.

[Liu et al. 2007] Liu, Y., Dion, F. and Biswas, S. (2007) 'Microscopic traffic simulation modeling considering vehicle-to-vehicle communication delays', *Proc. 18th IASTED International Conference*, Montreal, Canada, pp.556–561.

[Logan et al. 2001] Logan, B. and Theodoropolous, G. (2001) 'The Distributed Simulation of Multi-Agent Systems', *Proc. of the IEEE*, Vol. 89, No. 2, pp.174–185.

[Kumova 2005] Kumova, B.I. (2005) 'Dynamically Adaptive Partition-Based Data Distribution Management', *Proc. 19th workshop on Parallel And Distributed Simulation*, Monterey, California, pp.292–300.

[Misra 1986] Misra, J. (1986) 'Distributed discrete event simulation', *ACM Computing Surveys*, Vol. 18, No. 1, pp.39–65.

[Perrone et al. 2003] Perrone, L.F., Yuan, Y. and Nicol, D.M. (2003) ' Simulation of large scale networks II: modeling and simulation best practices for wireless ad hoc networks', *Proc. 35th Int'l Conf. on Winter Simulation*, New Orleans, Luisiana, pp.685–693.

[Peschlow et al. 2007] Peschlow, P., Honecker, T. and Martini, P. (2007) 'A Flexible Dynamic Partitioning Algorithm for Optimistic Distributed Simulation', *Proc. 21th ACM/IEEE/SCS International Workshop on Principles of Advanced and Distributed Simulation*, IEEE Press, San Diego, California.

[Raczy et al. 2005] Raczy, C., Tan, G. and Yu, J. (2005) 'A sort-based DDM matching algorithm for HLA', *ACM Trans. Model. Comput. Simul.*, Vol. 15, No. 1, pp.14–38.

[Shanaker et al. 2001] Shanaker, M.S., Padman, R. and Kelton, W.D. (2001) 'Efficient distributed simulation through dynamic load balancing', *IIE Transactions, Springer*, Vol. 33, No. 3, pp.203–217.

[Som et al. 2000] Som, T.K. and Sargent, G. (2000) 'Model structure and load balancing in optimistic parallel discrete event simulation', *Proc. fourteenth workshop on Parallel And Distributed Simulation*, Bologna, Italy, pp.147–154.

[Szymanski et al. 2002] Szymanski, B.K., Saifee, A., Sastry, A., Liu, Y., Madnani, K. (2002) 'Genesis: a system for large-scale parallel network simulation', *Proc. 16th workshop on Parallel And Distributed Simulation*, Washington DC, USA, pp.89–96.