Performance Analysis of a Parallel and Distributed Simulation Framework for Large Scale Wireless Systems

Luciano Bononi   Michele Bracuto   Gabriele D'Angelo   Lorenzo Donatiello

Dipartimento di Scienze dell'Informazione, Università degli Studi di Bologna,
Mura Anteo Zamboni 7, 40126, Bologna, Italy

{bononi, bracuto, gdangelo, donat}@cs.unibo.it

Abstract
The simulation of ad hoc and sensor networks often requires a large amount of computation, memory and time to obtain significant results. The parallel and distributed simulation approach can be a valuable solution to reduce the computation time, and to support model components' modularity and reuse. In this work we perform a testbed evaluation of a new middleware for the simulation of large scale wireless systems. The proposed middleware has been designed to adapt and to scale over a heterogeneous distributed execution infrastructure. To realize a testbed evaluation of the considered framework we implemented and investigated a set of wireless systems' models. Specifically, we identified two classes of widely investigated wireless models: mobile ad hoc, and static sensor networks. In this work we present the performances of the simulation framework, with respect to the heterogeneous set of execution architectures, and the modeled systems' characteristics. Results demonstrate that the framework leads to increased model scalability and speed-up, by transparently adapting and managing at runtime the communication and synchronization overheads, and the load balancing.

Categories and Subject Descriptors
I.6.8 [Simulation and Modeling]: Types of Simulation – discrete event, distributed, parallel

General Terms
Performance, Design, Experimentation

Keywords
Parallel and distributed simulation, wireless systems, middleware design, scalability, overheads reduction

1. INTRODUCTION
The research in the field of wireless systems, protocols and architectures has been characterized by the need to investigate even more complex and detailed models of wide, scalable and integrated systems. Many standards and system architectures for wireless systems have been proposed, and are currently being deployed and developed, while other solutions are currently under the design and analysis phases for future deployment. Wireless networks' architectures and wireless sensor systems are currently under analysis to obtain insights and guidelines governing many relevant design issues: as an example, system architecture and management choices, protocols' design, dynamic self-configuration and adaptation to system dynamics, systems and protocols' interoperability and co-existence, system scalability and system fault-tolerance and lifetime. To obtain valuable insights of the investigated indices, researchers would require intuitive, accurate and fine-grained methodologies and tools for the analysis. Because of mathematical untractability and the model complexity, simulation-based investigation of wireless systems is often preferred to numerical and analytical resolution methods [18, 28, 33, 34, 36]. Simulation makes more practical the modeling and investigation of complex and dynamic scenarios, often characterized by multiple correlated factors, "memory effects" of the system states, and dynamic causal effects. Under such conditions, a simulation model would allow a level of detail that exceeds the detail level that could be obtained in most tractable mathematical models. A modular simulation modeling makes it possible the model components' reuse and composition, and works in favour of a correct system design, together with the possibility of a preliminary functional-test and interoperability analysis that would result in a fast system deployment.

On the other hand, two problems appear to limit the adoption of simulation techniques for the analysis of complex systems: i) the limitation of affordable-cost simulation execution architectures (mainly memory and computational power) [28, 33, 34], and ii) the scarce possibility of model components' reuse and model composition among heterogeneous simulation models and tools. As a consequence, the research for tools and new methodologies for standard- and module-based modeling and simulation of large-scale and complex wireless networks has received a great attention by the research community, and has led to some interesting results [1, 2, 5, 7, 15, 16, 17, 19, 20, 22, 27, 29, 30, 35, 38]. Currently, it is widely recognized that many of the most adopted tools for simulation of wireless systems (e.g. Network Simulator, ns2 [37]) suffer the memory limitation of the execution architecture (which translates in a limitation of the model complexity and scalability). Also, they suffer the great computation time required to complete the simulation processes [26]. The fact that wireless networks models may be complex and may include a potentially huge number of simulated and

MSWiM'04, October 4–6, 2004, Venezia, Italy.
Copyright 2004 ACM 1-58113-953-5/04/0010...$5.00.

dynamically interacting components would represent an amplifier of the modeling limitations due to memory constraints of current tools and simulation architectures. Specifically, under the computation viewpoint, the simulation of wireless systems' models may require a long time, due to the execution of complex behaviors and state updates required on behalf of many model components. Many different model factors may introduce additional computation requirements for implementing detailed model components behaviors: e.g. mobility patterns and topology changes, layered communication protocols, resource sharing, interference effects, among others. As a result, large scale and complex simulation models are often unpractical to simulate on a single-processor execution unit, because of huge memory requirements and large amount of time required to complete the simulation runs [28, 33, 34]. Parallel and distributed models and architectures may be a viable alternative to reduce memory bottlenecks through distributed memory hierarchies, and to obtain simulation speed-up thanks to the parallel execution of computation tasks [1, 6, 7, 13, 14, 19, 20, 21,22, 31, 32, 34, 37, 40]. A parallel and distributed simulation is realized by a set of distributed execution units, where concurrent executions of parallel processes are controlled and synchronized basically via message-passing primitives. Such primitives are used to distribute event-messages and to implement distributed time management. Many distributed simulation paradigms can be adopted to realize a parallel and distributed simulation, whose illustration is out of the scope of this paper [13, 14].

Recently, a new standard has been defined, named IEEE 1516 Standard for parallel and distributed modeling and simulation [17]. The standard defines rules and interfaces allowing for heterogeneous model components' interoperability in parallel and distributed simulations. Model components (formally known as federates) are executed as Logical Processes (LPs) [17]. The standard, often (and improperly) referred to as High Level Architecture (HLA), defines a set of rules and service interfaces that can be implemented by a RunTime (RTI) middleware. The HLA rules must be followed, and the RTI services can be exploited, by every model component in the simulation [8, 12, 17]. The HLA implementations available so far have gained a good interest and diffusion, but have also been subject to some criticisms [8, 12, 39, 3, 4, 10]. The investigation of new features and services, and the lack of Open Source runtime implementations are the main motivations behind the design and implementation of the ARTÌS (Advanced RTI System) middleware, which we illustrate and test in this paper.

ARTÌS is a new parallel and distributed simulation middleware, whose aim is to support the simulation of complex, massively populated systems, providing a simple and straightforward set of services available to the simulation modelers. As many other middlewares, ARTÌS is oriented to support model heterogeneity, distribution and reuse. On the other hand, the ARTÌS design was focused on the exploitation of assumptions and characteristics of both the execution system architecture, and the simulation model, to increase the performances in parallel and distributed simulations [4].

The main bottleneck arising in a distributed simulation framework is given by the communication overheads to realize the event-message distribution and synchronization services between a set of distributed model entities. The communication overhead due to the message passing required for the parallel and distributed simulation could nullify all the performance gain obtained by parallel executions. The ARTÌS framework is designed to realize the dynamic adaptation of the interprocess communication layer to the heterogeneous communication support offered by possible different simulation-execution architectures [4]. Specifically, we oriented the ARTÌS design towards the adaptive evaluation of the communication bottlenecks and support for multiple communication infrastructures and services, from shared memory to Internet-based communication [4]. ARTÌS has been also integrated in a framework with another middleware, named Generic Adaptive Interaction Architecture (GAIA). Basically, GAIA implements a simple model components' migration mechanism, preliminarly proposed on the top of HLA-based distributed simulations [3]. The HLA standard and existing RTIs do not define component migration facilities, and preliminary research activity was made on this topic [24, 25]. GAIA includes a heuristic migration policy, whose aim is to dynamically partition and allocate the interacting model components over many Logical Processes (LPs), respectively executed over a set of multiple, distributed execution units. The composition of ARTÌS and GAIA realizes a framework for parallel and distributed simulation, characterized by adaptive reactions to dynamic systems' behavior, and oriented to the communication-overhead reduction. In this work, the prototype implementation of the ARTÌS and GAIA framework is outlined. A set of testbed-evaluation results are presented to express the potential of ARTÌS with and without GAIA over heterogeneous execution architectures, and for two classes of wireless systems' models: wireless mobile ad hoc networks and wireless sensor networks.

The paper structure is the following: in section 2 we outline the guidelines and motivations for the modeling and implementation of distributed simulation of wireless systems. In section 3 the ARTÌS middleware and the GAIA framework are introduced. In section 4, two wireless system's models are described. In section 5 we present a set of simulation results. In section 6 we summarize our conclusions and future work.

2. DISTRIBUTED SIMULATION OF WIRELESS SYSTEMS

Wireless systems are highly dynamic systems where the interactions are subject to fast changes driven by the system evolution. Given a wireless system model, it is quite natural for modelers to implement and recycle a set of model components, each one realized by a composition of software modules, to obtain the global system model. Every autonomous model component (e.g. a wireless node or sensor) is then required to mimic the interactions (i.e. the causal effects of events) with all the neighbor components in the modeled scenario. Two inherent characteristics of wireless systems play an important role in the modeling and simulation viewpoint: i) the hosts' mobility and ii) the open broadcast nature of the wireless transmissions.

As an example, topology changes due to simulated hosts' mobility map on causality effects in the "areas of influence" of each mobile device. This may result in dynamically shaped causality-domains, which map on the interaction-scheme of the distributed model components. Intuitively, given two or more neighbor-hosts sharing the wireless medium, the causal effect of a signal-interference event due to the "open broadcast" nature of the wireless transmissions could result in a chain of local-state events up to the high protocols' layers [36]. In our modeling approach, we define a model entity as the data structure defined to model a Simulated Mobile Host (SMH).

A high degree of causality in the simulation of the wireless hosts' communication is driven by the local-topology interaction (i.e. transmissions) between neighbor hosts [18, 36]. Under the modeling and simulation viewpoint, if a SMH changes its position, it will eventually interact with a new community of neighbor hosts. A certain degree of time-locality among neighbors' communications can be considered an acceptable assumption in many wireless system models, depending on the communication load and the mobility model assumptions. The system and model dynamics can be influenced by motion model and speed, and also by the SMHs density.

To realize a correct evolution in a parallel or distributed simulation, under the event-causality viewpoint, every model components' interaction should be notified as an event-message to all the causally dependent model components. In a distributed simulation, this task is usually managed by a runtime event-message distribution mechanism. Complex systems with detailed and fine-grained simulation models can be considered communication-intensive under the distributed simulation approach. As a result, interprocess communication may become the bottleneck of the distributed simulation paradigm. The way interprocess communication can be supported in distributed systems would mainly depend on the execution units and on the simulation system resources, architectures and characteristics. As an example, message passing communication can be performed efficiently over shared memory architectures, while it would require medium/high communication latencies over local and wide area network communication services.

The physical clustering of interacting model components on a shared memory architecture could result in the advantage to exploit the most efficient message passing implementation. Unfortunately, in wireless mobile networks any optimal, static clustering and allocation of model entities, based on the current component-interaction scheme, will become immediately suboptimal, due to the dynamics of the model interactions (e.g. SMH mobility). The approach used in currently available implementations of parallel and distributed simulation frameworks is to passively detect the model component interactions, by adapting the event message distribution accordingly. No background optimization is based on the heterogeneous characteristics of any available communication infrastructure. This observation will be a motivation for the design of our simulation framework.

The event-message distribution of a distributed simulation requires a dynamic definition of publishing/subscribing lists, or the implementation of a complete state-sharing information system. On the other hand, a dynamic approach for the event-distribution and state-information-updates (e.g. dynamic lists and subscription groups) would lead to additional communication and management overheads. In some scenarios, the communication cost of list-updates or fine-grained events' communication between a dynamically variable set of components, could make attractive a complementary approach. As an example, when the system communication infrastructure is characterized by significant performance asymmetry (e.g. shared memory vs. LAN communication), like in networked clusters of PCs, the migration cost needed to dynamically cluster the set of interacting components over a single Physical Execution Unit (PEU) could become attractive. This would be even more attractive if the following three assumptions could be satisfied: i) components' migration could be implemented incrementally as a simple data-structure (i.e. state) transfer, ii) the component state would be comparable with the amount of data exchanged for interactions, and iii) the object's interaction scheme would be maintained for a significant time (i.e. time-locality).

2.1. The Simulation Execution Testbed

Our simulation testbed consists of a distributed, discrete-event simulation of model components. Model components are executed as logical processes over a set of physical execution units (PEUs), connected by a physical LAN network.

The execution architecture for our experiments is realized by 3 PEUs each one equipped by Dual Xeon Pentium IV 2800 Mhz, with 3 GB RAM, and one PEU equipped by Quadral Xeon Pentium IV 1500 Mhz, with 2GB RAM, all connected by a Fast Ethernet (100 Mb/s) LAN, and all equipped with Debian GNU/Linux OS with kernel version 2.6.

Our design approach is mainly focused on the adaptive communication-reduction between the PEUs where Logical Processes (LP) are executed. Every LP is statically allocated and executed on a single PEU. Specifically, one single LP cannot be split over two or more PEUs, more LPs can be executed over a single PEU, and LPs cannot be migrated between PEUs.

Every LP is managed by a runtime simulation core (ARTÌS) as a single simulation component. On the other hand, a single LP is implicitly formed by a set of threads, each one managing and updating the state (i.e. local data structures) of a set of Simulated Mobile Hosts (SMHs). A communication between wireless hosts can be modeled as a set of interactions (i.e. message-events) between any couple of adjacent SMHs. Since a wireless communication must be always modeled as a broadcast within a limited local transmission range, this requires that each SMH within a variable range would be notified with the transmission-related event-messages. Each event would result in a multiple set of one-to-one interactions (i.e. event messages) among local SMHs. If the sender SMH and its neighbors belong to the same LP (i.e. they are executed on the same PEU), or if they belong to different LPs implemented over the same PEU, then their interactions can be considered local (e.g. shared memory communication) and do not involve any physical network communication. On the other hand, every interaction involving participants implemented over foreign LPs (e.g. LPs implemented over different PEUs) may require time-expensive physical network communication. By reducing the physical network communication we can reduce the synchronization delays. By clustering neighbor SMHs within the same LP, or within the LPs executed over the same PEU, we close the causal interactions and system communication within the PEU where the interacting LPs (and their respective SMHs) are executed. In addition, clustered interacting SMHs would limit interactions with the management layers of the ARTÌS middleware, by further reducing the computation and communication overheads. To sum up, by limiting the network communication in favour of the local (shared memory) communication, the wall clock time required by the simulation runtime to achieve full synchronization would be reduced. This would make it possible to obtain a simulation speedup.

A static approach could be adopted to optimally distribute the SMHs within the LPs in the simulation initialization phase. The optimal solution for allocation is hard to find and could be defined in many ways, depending on the targeted overheads' reduction. Typically, the optimality is defined with respect to latency (to reduce the physical network communication cost) or computation (to obtain an optimally balanced execution parallelism). Anyway, this should be explicity performed offline by the modeler, on the basis of the modeling assumptions. Moreover, as it will be demonstrated in the final results, the model dynamics (e.g. the SMH mobility) would make the initially optimal distribution less effective after few simulation steps. This result may translate in a performance degradation for the simulation speedup, mainly due to the increasing cost of communication and synchronization required between distributed model components (logical processes). In our approach the optimization is dynamically performed at runtime, by the proposed simulation middleware, by migrating the SMHs between LPs. In this way, the modeler is relieved by the optimization task, and the system converges towards a balanced, tuneable and pseudo-optimal model components' distribution driven by the model interaction scheme. If we assume a time-locality in the interaction between neighbor hosts, it could be convenient to migrate the foreign SMH to the LP (and to the PEU) where its new neighbors are located, by reducing in this way the cost of successive interactions. This assumption is typically verified in MANETs, e.g. most routing protocols are based on "proximity" concept to decide the routing path of communications, and such communications usually last for a significant time, following a bidirectional session-based scheme. The effect of the time-locality of the causality effect inside each logical process will be investigated in the final section, by varying the SMH mobility speed.

3. THE DISTRIBUTED SIMULATION FRAMEWORK

3.1. The Advanced RTI System (ARTÌS)

The main purpose of ARTÌS is the efficient support of complex simulations in a parallel and distributed environment.

The ARTÌS implementation [4] follows a component-based design, that results in easily extendable middleware (see figure 1). The solutions proposed for time management and synchronization in distributed simulations have been widely analyzed and discussed in the design phase. Currently, ARTÌS supports the conservative time management based on both the time-stepped approach, and the Chandy-Misra-Bryant algorithm. We are working on the extension of ARTÌS to support optimistic time management algorithms. The initial choice to support the conservative approach was a speculation on the highly unpredictable characteristics of wireless system models [3], which may result in frequent rollbacks under optimistic simulation schemes. In ARTÌS, many design optimizations have been applied to the communication layer, to obtain adequate protocols for synchronization and communication in Local Area Network (LAN) or Shared Memory (SHM) multiprocessor architectures (see figure 1). In our vision the communication and synchronization middleware should be adaptive and user-transparent about all the optimizations required to improve performances. The current scheme adopts an incremental straightforward policy: given a set of LPs on the same physical host, such processes always communicate and synchronize via read and write operations, performed within the address space of LPs, in the shared memory. To implement these services we have designed, implemented and tested many different solutions, based on Inter Process Communication (IPC) semaphores and locks, busy-waiting, and "wait on signals" with a limited set of temporized spin-locks. The latter solution has demonstrated very low latency and limited CPU overhead, good performances obtained in multi-CPU systems, good scalability, and no need to reconfigure the operating system kernel level.

Two or more LPs located on different hosts (i.e. no shared memory available), on the same local area network segment, communicate by using a light Reliable-UDP (R-UDP) transport protocol over the IP protocol. Ongoing activity is evaluating the use of raw sockets for R-UDP data segments directly encapsulated in MAC Ethernet frames (i.e. bypassing the IP layer). Two or more LPs located on Internet hosts rely on standard TCP/IP connections.

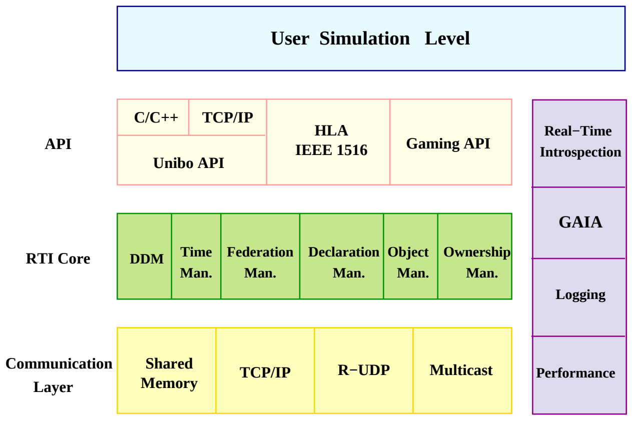

Figure 1: The ARTÌS framework modules and architecture

3.2. The Generic Adaptive Interaction Architecture (GAIA)

The PDES simulator built to obtain an experimental testbed of our proposal is based on a distributed architecture made by a set of logical processes glued together by the ARTÌS middleware. In our preliminary design [3] we adopted the High Level Architecture (HLA) DMSO (Department of Military Simulation Office, US Department of Defense) implementation RTI-1.3NGv3.2 as the basis for our work. On top of the HLA RTI we built a middleware extension called Generic Adaptive Interaction Architecture (GAIA). GAIA provides the interaction to the simulation core, the location and distribution data management, the random number generator, tracefile-logging and other simulation facilities.

The target of GAIA is to provide migration and service APIs to the simulation developer. Because of the unavailability of DMSO RTI source-code, the GAIA facilities were initially provided as an external middleware on top of the DMSO RTI [3]. The development of ARTÌS middleware has permitted to merge the GAIA framework within the runtime core, still reducing the runtime execution overheads (see figure 1).

We implement SMH models as code with data structures to define and maintain the SMH state information. GAIA migrates the "data structure", i.e. the state information of SMHs between LPs. This required to design and to implement a migration layer for the "state" of the SMH model entities between LPs. The ARTÌS runtime has been extended to execute static models and to exploit migration by means of a small set of Application Programming Interfaces (APIs) providing migration services for migration-enabled models.

To test our framework we implemented a time-stepped, conservative, parallel and distributed discrete-event simulation of two classes of models for mobile wireless systems.

It is worth noting that the ARTÌS and GAIA implementation under test in this work are completely different than the implemented tools preliminary analyzed in [3]. By following the guidelines obtained in [3], the ARTÌS runtime has been designed and implemented as an alternative to HLA runtime, and the GAIA middleware has been completely reimplemented, with both the migration and the load-balancing heuristics completely redesigned. Moreover, the composition of GAIA with ARTÌS results in lower management overheads and greater speedup than the framework architecture described in [3].

4. WIRELESS SYSTEMS' MODEL DEFINITION

In the following we describe two classes of wireless systems' models that we considered in our testbed evaluation of the ARTÌS and GAIA framework. The two classes of models have been selected i) because they represent two examples of widely studied systems and ii) because they capture most of the complementary characteristics of wireless simulation systems about mobility, communication load, management overheads, resource limitations. This will let us to obtain results about the optimization and speedup achieved by our simulation framework, based on the exploitation and adaptation to many variable model characteristics.

4.1. The Mobile Ad Hoc Network's Model

The definition of a mobile ad hoc network model is basically devoted to study the effect of hosts' mobility and high communication loads assumptions under the modeling viewpoint. We assume a highly scalable number of simulated mobile hosts (SMHs), each one following a Random Mobility Motion model (RMM). This motion model is syntetic and far from reality, but the choice was driven by the unpredictable and uncorrelated mobility pattern of SMHs. This is the worst case analysis for our mechanism, because any heuristic definition cannot rely on any assumption about the motion correlation and predictability of SMHs. The only correlation effect we would exploit in our mechanism is given by the "time-locality" of communication sessions between neighbor-hosts. Given the framework definition, our feeling is that any other widely used motion model, like any restricted, correlated or Group Mobility model, would result in better results than the adopted RMM model, for any migration heuristic. The RMM model is defined by SMHs swinging between mobile and static epochs. At the beginning of each epoch, every SMH decides to stay or to change its mobile or static status, by following a geometric distribution with parameter p=1/2. When entering a mobile state, new, uncorrelated and uniformly-distributed direction and speed are randomly selected and maintained up to a static epoch. The cycle is repeated for the whole simulation run by every SMH. Sometimes we considered motion sub-models related to the motion speed, i.e. high speed (25 spaceunits/timestep), and lower speed (10 spaceunits/timestep). To stress the migration scheme, we have also used an extreme sub-model with very high speed (100 spaceunits/timestep).

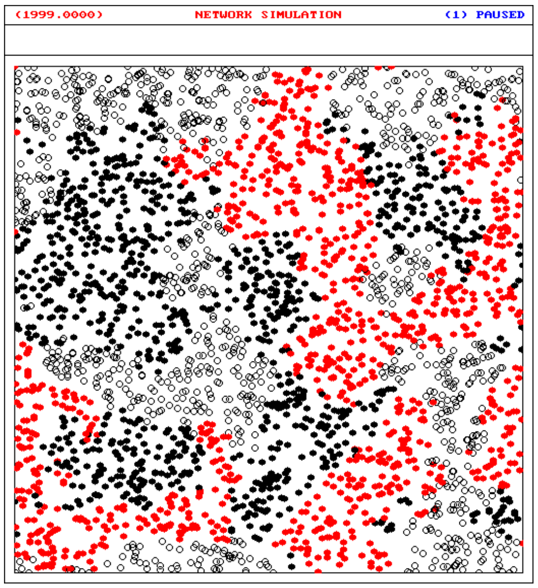

Figure 2: a snapshot of a Mobile Ad Hoc Network with 3000 SMHs dynamically allocated by GAIA over 3 PEUs. Dot colors define the PEU where the SMH is executed.

Space is modeled as a torus-shaped 2-D grid-topology, 10.000x10.000 spaceunits, populated by a constant number of mobile SMHs. SMHs are randomly and uniformly distributed in the simulated area (see dots positions in figure 2). The torus space topology, indeed unrealistic, is commonly used by modelers to prevent non-uniform SMHs' concentration in any area. This allows to evaluate the mechanism behavior in a worst case scenario, where the clustering of SMHs is not trivially determined by high concentration in small areas. We believe that these are stressing examples for our mechanisms, because they will lead to a high migration overhead, given the motion model defined. The simulated space is wide and open, without obstacles. The modeled communication between SMHs is a constant flow of ping messages (i.e. constant bit rate), transmitted by every SMH to all neighbors within a wireless communication range of 250 spaceunits. Again, this choice is stressing the migration mechanism under the mobility effects of continuously transmitting SMHs. In our proposal, since the SMH migration policy is evaluated on the basis of the local and remote interaction (i.e. communication), no communication translates in no migration needs, hence no additional communication, synchronization and migration overheads. The rate of ping messages is constant because it is the control parameter for communication: increasing/reducing the ping rate would be equivalent to change the interaction rate. We plan to extend this model with the real implementation of message flows, routing protocols and applications as a future work. Currently, the host model implements the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) Medium Access Control protocol of the IEEE 802.11 Distributed Coordination Function. Anyway, this model is not used in our investigation because we are only interested in modeling the basic interaction, at the physical layer, which is given by the event of a channel occupancy due to local ping messages among neighbor hosts. It is worth noting that additional local computation would be required by adopting more detailed and complete protocol stacks implementations over the SMH model entities, resulting in additional advantages of parallel execution.

4.2. The Sensor Network Model

The second model we considered in our testbed evaluation is based on a wireless sensor network system. In this model we are interested to test the model scalability, by showing results in a system with up to 40.000 sensors. The most important feature of this system with respect to the mobile ad hoc network, is given by the fact that sensors are static. They are randomly placed with uniform distribution in the simulated area (see figure 3).

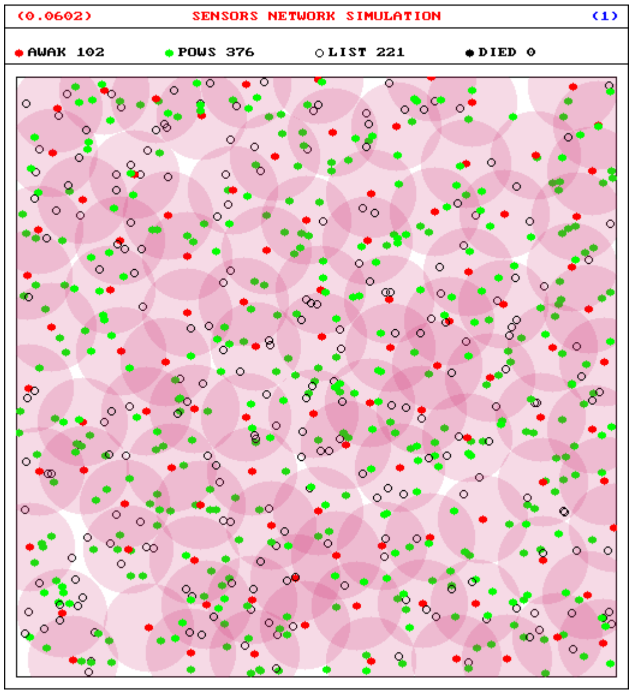

Figure 3: a snapshot of a wireless sensor network with 700 sensors. Dot colors refer to sensor state: red=active, green=power saving, white=listening.

For maintaining the protocol behavior and average connectivity, the area size is variable such that the sensor density is constant in all experiments (approximatively 1 sensor in 10x10 space units). The communication range of each sensor is 15 space units. Every sensor implements a "pressure variation" detector and sends broadcast alerts that are flooding towards a set of target detection points. To maximize the network lifetime, every sensor implements a power saving mechanism that adaptively manages the sensor state. A sensor can be active, listening and in power saving state. A new Medium Access Control scheme, whose definition is out of the scope of this paper is currently investigated by adopting the model defined. Under the modeling and simulation viewpoint this model is complete and provides detailed information about the system behavior, both for sensor network management, communication, resources' utilization and network lifetime indices.

5. EXPERIMENTAL RESULTS

In this section we present the results of some testbed simulation experiments executed to test the ARTÌS distributed simulation framework, and the GAIA middleware. The experiments have been performed over heterogeneous execution infrastructures and scenarios, and have involved two different classes of wireless systems models. The motivation for this study is given by the evaluation of the adaptive self-configuration of the ARTÌS framework and the GAIA middleware, executed in transparent way with respect to the model and execution infrastructure characteristics. The target indices to be evaluated include: the observation of migration overheads related to model dynamics under GAIA, the advantages obtained by GAIA and ARTÌS under the adaptive communication overheads reduction, and the speedup obtained by our framework under variable execution scenarios and under variable modeling assumptions for the simulated wireless systems.

The execution architecture for our experiments was described in section 2.1. We performed multiple runs of each experiment, and the confidence intervals obtained with a 95% confidence level are lower than 5% the average value of the performance indices shown.

In the following we define as M the number of physical execution units (PEUs) supporting the simulation execution, and as N the total number of logical processes (LPs) implemented. With "migration ON" or "migration OFF" we identify a distributed simulation with the GAIA migration heuristic turned ON and OFF, respectively. All the performed experiments were started with a pseudo-random, uniform distribution of a variable number of SMHs for both the ad hoc, and the sensor networks models. Initially, the set of SMHs is randomly allocated over the set of PEUs, without any optimal allocation. The choice of the initial random distribution allows to analyze the transient dynamic effect of the GAIA migration mechanism. In [3] we shown that the random distribution would be asymptotically obtained if migration is disabled, starting from any initial (and optimal) allocation scheme, due to the SMHs' mobility. Most of the figures presented show transient behavior of the performance indices, because this describes the dynamics and fast convergence effect of the proposed mechanisms. Steady-state results have been also discussed to define the asymptotical behavior of the proposed framework.

5.1 Mobile Ad Hoc Network's Simulation

Figure 4 shows the transient number of model components (SMH) migrations performed by the GAIA middleware during the initial phase of a distributed simulation of the mobile ad hoc network model. The model is composed by 5000 SMHs, randomly distributed in the simulated area, and randomly allocated over three PEUs. The migration heuristic of GAIA begins to migrate the SMH model components between PEUs after a warmup (observation) phase of 50 timesteps. The figure 4 shows the transient number of migrations performed between the three PEUs in every timestep, based on the average speed value of SMHs in the simulated mobile ad hoc network (i.e. 10, 25 and 100 m/s). The SMH speed here is just a modeling factor to stress the simulation and it is not expected to be realistic. It can be observed that in the initial phase the GAIA middleware induces a peak of model components reallocation aiming to clusterize the interacting SMHs over the same PEU. The resulting model component allocation over PEUs at timestep 1000 would be similar to the distribution shown in figure 2. In figure 2, one dot represents a SMH in the simulated area, and the color of dots indicates the PEU where the SMH is executed. It is clear the clustering effect obtained by GAIA migrations after the initial transient phase. At the steady state, the SMH (dots) mobility would smoothly require a continuous adaptation and migration of SMHs moving out of the context of SMHs executed over the local PEUs. Figure 4 indicates that the higher the mobility (speed) of SMHs, the higher is the reallocation rate required by GAIA to optimize the degree of local communications within the PEUs.

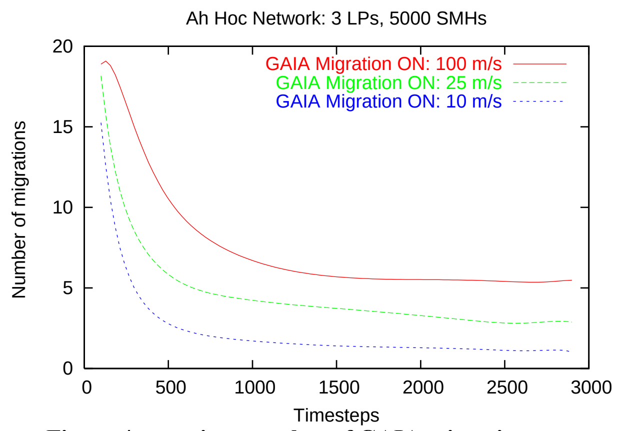

Figure 4: transient number of GAIA migrations per timestep, with respect to modeled mobility parameters

Figure 5 shows the Local Communication Ratio (LCR) of messages originated by the simulation in the same scenario considered in figure 2 and 4. The LCR is intuitively defined as the percentage of message passing required by the simulation execution which is local to each one of the three PEUs adopted for the execution. Given the ARTÌS design and assumptions, local message passing translates in efficient shared memory communication within each PEU, as an alternative to less efficient and time consuming network communications. Figure 5 shows that the GAIA migration allows to obtain a steady-state percentage of local communications around 85% in the considered scenario. Figure 5 also indicates that the mobility of SMHs have less or no effect on the LCR index when GAIA migration is active. This is due to the adaptive effect of GAIA migrations at runtime, shown in figure 4. The same simulation scenarios with GAIA migration OFF result in a LCR index which is around 33%, as expected when interacting SMHs are randomly allocated over three PEUs.

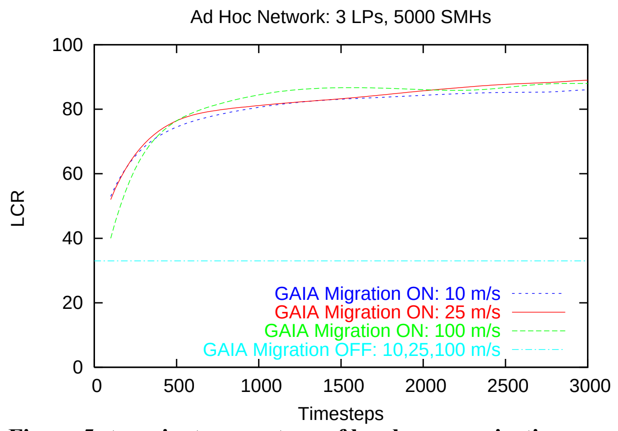

Figure 5: transient percentage of local communications per timestep, with respect to modeled mobility parameters, with and without GAIA migration

Figure 6 shows results about the speedup of the distributed simulation under the mobile ad hoc modeling scenario considered above. The speedup is shown as a transient index, by averaging the consecutive speedup indices calculated over separated and adjacent simulated time windows. This allows to evaluate the transient effect of the speedup in the initial phase, and when GAIA is switched OFF at runtime. The monolithic scenario is considered as the normalization value for speedup evaluation. A monolithic simulation is intended as a single logical process (LP) executed over a single PEU. In our implementation, when analyzing the simulator performances, we considered the monolithic execution platform as equivalent to a single sequential simulator. We realize that this assumption is not true in practice, because a really small biasing effect is introduced by the ARTÌS middleware in background, anyway the biasing is really low, and this assumption allows us to study comparable systems and models.

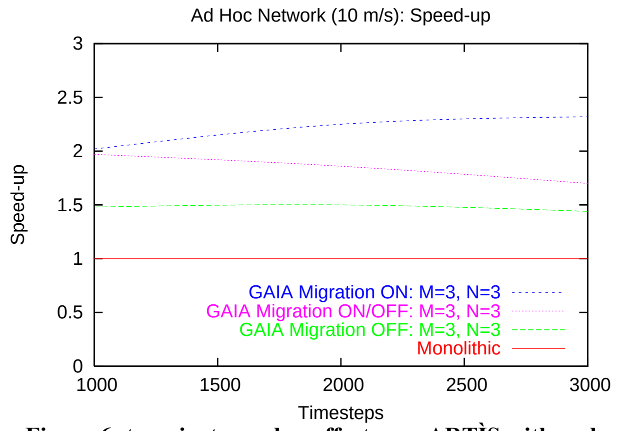

Figure 6: transient speedup effect over ARTÌS with and without the GAIA migration mechanism.
Average SMH speed: 10 m/s

When the simulation is executed over 3 PEUs (M = 3) and each PEU implements a single LP (N=3) with GAIA migration OFF, the speedup obtained is around the value 1.5 with respect to the monolithic execution scenario. In the same scenario with GAIA migration ON, the speedup starts around the value 2 and it increases up to 2.3 by the effect of GAIA dynamic reallocation and the increase of local communications. The curve labeled “GAIA Migration ON/OFF” shows the effect of degradation of the speedup obtained when, after the initial reallocation of GAIA, the GAIA migration is switched off: it is clear how the dynamic effect of SMH mobility (whose average speed is 10 m/s) realizes a transient mutation of interactions (i.e. communication) from local to external for the PEUs, by decreasing the speedup.

Figure 7 shows the same indices of figure 6, with the only difference given by the average speed of the modeled SMHs: from 10 m/s in figure 6 to 25 m/s in figure 7. As expected, the high modeled speed translates in less “time-locality” effect of local interactions. This reduces a little the speedup index obtained, because GAIA introduces less local communication advantages. On the other hand, the relative differences among the considered scenarios and mechanisms remain valuable, as in previous case. The same consideration about “time-locality” can be applied to explain why the speedup degradation when the GAIA migration is switched ON/OFF at runtime is faster in figure 7 than in figure 6.

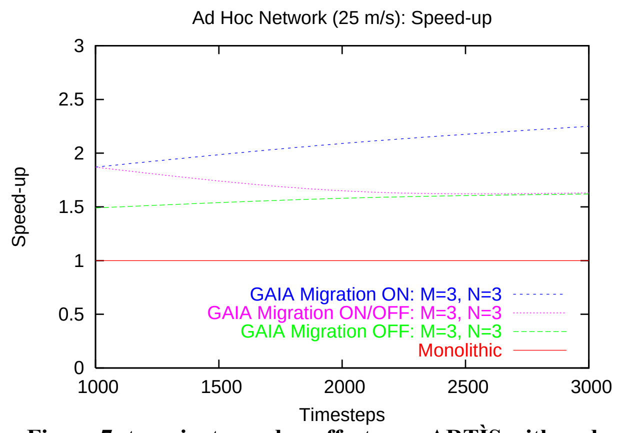

Figure 7: transient speedup effect over ARTÌS with and without the GAIA migration mechanism. Average SMH speed: 25 m/s

Figure 8 shows the speedup investigation of many execution system architectures, based on the mobile ad hoc network model characterized by 5000 SMHs with average speed of 25 m/s. Every bar in the histogram shows the speedup with respect to the monolithic implementation, with GAIA Migration Off and On, respectively. The first couple of bars on the left are just a reference of the monolithic normalized speedup. In general, the GAIA migration has a positive effect on the speedup indices, with many PEUs, by increasing the speedup indices up to 25%. More specifically, the second couple of bars shows the speedup obtained by 3 LPs over 1 PEU. This indicates that ARTÌS is able to exploit the shared memory, dual processor architecture of the PEU, when implementing the simulation splitted on 3 LPs. In the same way, as reported on the histogram, by increasing the number of PEUs (M value) in the execution architecture, ARTÌS and GAIA show to scale and to support more LP executions by gaining simulation speedup. In addition, in all the execution scenarios, GAIA dynamically recovers the insorgence of network communication due to model assumptions (SMH mobility), resulting in additional speedup.

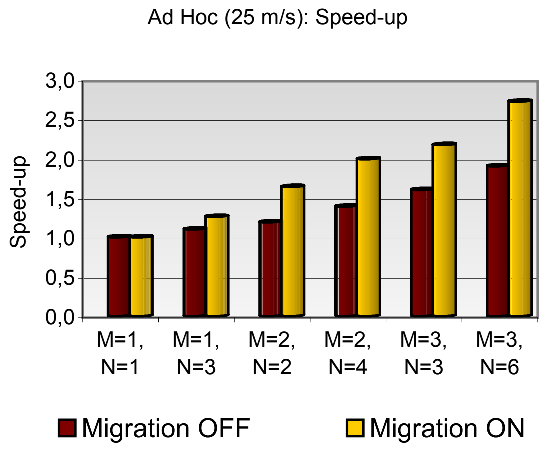

Figure 8: speedup investigation of ARTÌS and GAIA over many execution system architectures

5.2 Sensor Network’s Simulation

Figure 9 shows the scalability and speedup obtained by the simulation of the sensor network model previously defined. The effect of GAIA here is not considered because we are testing the scalability of the model and simulation implementation for a model composed by static wireless sensors. Again, every PEU considered here is a shared memory, dual processor architecture. We considered 3 execution scenarios. The monolithic scenario is realized by one LP over one PEU. The scenario labeled (M=3, N=6) is realized by 3 PEUs connected by the Ethernet LAN, each one with 2 LPs executed over the dual processor, shared memory architecture. The scenario labeled (M=2, N=4) is realized by 2 PEUs connected by the Ethernet LAN, each one with 2 LPs executed over the dual processor, shared memory architecture. The figure shows the speedup obtained as a function of the number of modeled sensors, i.e. the ARTÌS speedup and scalability under the model complexity viewpoint. The speedup obtained increases with the number of simulated sensors. This can be explained because a huge number of sensors could exploit the potential for parallel computation expressed by the multiple number of processors in the execution architecture. The speedup obtained by 3 PEU outperforms the speedup obtained with only 2 PEUs, as expected. When the number of sensors is really high (i.e. around 40.000) the speedup index reaches the top value, and does not show reductions, indicating the good scalability achieved by the simulator performance. As a marginal note, by considering that a state occupancy of a sensor model entity in our experiments was around 250 Bytes, we executed a single experiment for a simulation of 1.000.000 sensors without having evidence of any problem. Additional investigation will be performed on the evaluation of GAIA reallocation mechanism under the sensor network scenario, in the initial phase. This would contribute to further optimize and increase the local communication and speedup obtained.

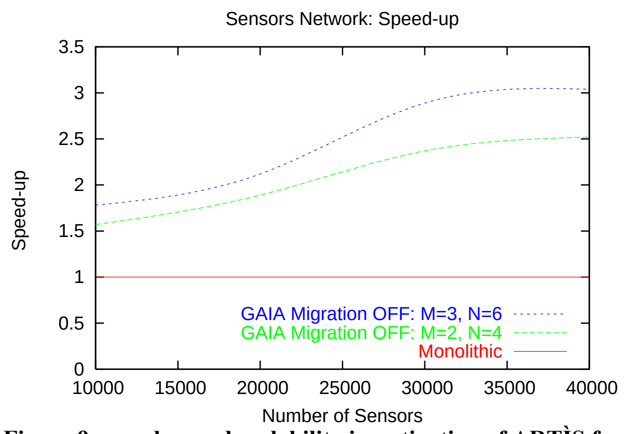

Figure 9: speedup and scalability investigation of ARTÌS for a massive sensor network model

6. CONCLUSIONS AND FUTURE WORK

In this work we illustrated the design and motivations, and we tested a new framework obtained as the integration of two recently developed middlewares defined to support the parallel and distributed simulation of large scale, complex wireless system models. The ARTÌS middleware is a new framework incorporating a set of features that allow an adaptive optimization of the communication layer management in a distributed simulation scenario supported by heterogeneous systems and communication services. ARTÌS has been integrated with GAIA, a dynamic mechanism for the runtime management and adaptive allocation of model entities in a distributed simulation. GAIA adapts the dynamic and time-persistent causal effects of model interactions to dynamic migration of model entities.

Two classes of wireless systems models have been considered in our testbed evaluation of the ARTÌS and GAIA framework. The two classes of models have been selected i) because they represent two examples of widely studied systems and ii) because they capture most of the complementary characteristics of wireless simulation systems. The definition of a mobile ad hoc network model was basically selected to study the effect of hosts' mobility and high communication loads assumptions under the model viewpoint. The static sensor network model was adopted to test the middleware scalability based on the ARTÌS runtime in isolation (i.e. without GAIA). This let us to obtain results about the optimization and speedup of our simulation framework, based on the exploitation and adaptation to variable model characteristics.

Results shown that the proposed simulation framework is able to transparently adapt to the execution system and model characteristics, by dynamically reducing the communication overheads, and by increasing the simulation scalability and speedup.

Our future work will include the ARTÌS extension with optimistic management and the definition of new models for dynamically interacting systems like multi-agent systems, P2P models, complete protocol stacks for ad hoc and sensor models, biology-inspired models and molecular systems, elementary particles physics and cosmology systems.

Acknowledgments

This work is supported by MIUR FIRB funds, under the project: "Performance Evaluation of Complex Systems: Techniques, Methodologies and Tools".

7. REFERENCES

[1] R. Bagrodia, R. Meyer, M. Takai, Y. Chen, X. Zeng, J. Martin, and H.Y. Song, "PARSEC: a parallel simulation environment for complex systems", IEEE Computer, 31(10), October 1998, pp.77-85

[2] A. Berrached, M. Beheshti, O. Sirisaengtaksin, and A. Korvin, "Alternative Approaches to multicast group allocation in HLA data distribution", Proc. Of the 1998 Spring Simulation Interoperability Workshop, 1998

[3] L.Bononi, G.D'Angelo, L.Donatiello, "HLA-Based Adaptive Distributed Simulation of Wireless Mobile Systems", in Proceedings of IEEE/ACM Intern. Workshop on Parallel and Distributed Simulation (PADS'03), San Diego, CA, June 2003

[4] L.Bononi, M. Bracuto, G. D'Angelo, L. Donatiello, "ARTÌS: a Parallel and Distributed Simulation Middleware for Performance Evaluation", University of Bologna Int. Report, http://www.cs.unibo.it/~bononi/Reports/ARTISTR.pdf, Mar. 2004

[5] A. Boukerche, and S.K. Das, "Dynamic Load Balancing Strategies for Conservative Parallel Simulation", Proc. of 11-th Workshop on Parallel and Distributed Simulation (PADS'97), June 1997, Lockenhaus, Austria, pp. 20-28

[6] A. Boukerche, S.K. Das, A. Fabbri, "SWiMNet: A Scalable Parallel Simulation Testbed for Wireless and Mobile Networks", ACM/Kluwer Journal on Wireless Networks, Vol 7, No 5, pp. 467-486. 2001

[7] A. Boukerche, and A. Fabbri, "Partitioning Parallel Simulation of Wireless Networks", Proc. of the 2000 Winter Simulation Conference (WSC), 2000

[8] J. Dahmann, R.M. Fujimoto, and R.M. Weatherly, "High Level Architecture for Simulation: an update", Winter Simulation Conference, December 1998

[9] S.R. Das, "Adaptive protocols for Parallel Discrete Event Simulation", Proc. of Winter Simulation Conference, 1996

[10] W.J. Davis, G.L. Moeller, "The High Level Architecture: is there a better way?", proc. Winter Simulation Conference, 1999

[11] E. Deelman, and B.K. Szymanski, "Dynamic load balancing in parallel discrete event simulation for spatially explicit problems", Proc. of the 12-th workshop on Parallel and distributed simulation PADS'98, July 1998

[12] DMSO: Defence Modeling and Simulation Office (1998), High Level Architecture RTI Interface Specification, Vers. 1.3

[13] A. Ferscha, “Parallel and Distributed Simulation of Discrete Event Systems”, In Handbook of Parallel and Distributed Computing, McGraw-Hill, 1995

[14] Fujimoto, R.M., Parallel and Distributed Simulation Systems, John Wiley & Sons, 2000

[15] B.P. Gan, Y.H. Low, S. Jain, S.J. Turner, W. Cai, W.J. Hsu, and S.Y. Huang, “Load balancing for conservative simulation on shared memory multiprocessor systems”, Proc. of the 14-th workshop on Parallel and distributed simulation (PADS’00), May 28-31, 2000, Bologna, Italy, p.139-146

[16] P. Huang, D. Estrin, and J. Heidemann, “Enabling large-scale simulations: Selective abstraction approach to the study of multicast protocols”, proc. Mascots'98, Oct. 1998

[17] IEEE Std 1516-2000: IEEE standard for modeling and simulation (M&S) high level architecture (HLA) - framework and rules, - federate interface specification, - object model template (OMT) specification, - IEEE Recommended Practice for High Level Architecture (HLA) Federation Development and Execution Process (FEDEP), 2000

[18] Internet Engineering Task Force, MANET WG Charter, http://www.ietf.org/html.charters/manet-charter.html

[19] K.G. Jones, and S.R. Das S.R., “Parallel Execution of a sequential network simulator”, Proc. of the 2000 Winter Simulation Conference, 2000

[20] O.E. Kelly, J. Lai, N.B. Mandayam, A.T. Ogielski, J. Panchal, R.D. Yates, “Scalable parallel simulations of wireless networks with WiPPET: modeling of radio propagation, mobility and protocols”, Mobile Networks and Applications, v.5, n.3, September 2000, pp.199-208

[21] M. Liljenstam, R. Ronngren and R. Ayani, “MobSim++: Parallel Simulation of Personal Communication Networks“, IEEE Distributed Systems, vol 2, No 2, February 2001

[22] W.W. Liu, C.C. Chiang, H.K. Wu, V. Jha, M. Gerla, and R. Bagrodia, “Parallel simulation environment for mobile wireless networks”, Proc. of Winter Simulation Conference, 1996

[23] B. Logan, and G. Theodoropoulos, “The Distributed Simulation of Multi-Agent Systems”, Proc. of the IEEE, 2001

[24] J. Lüthi, and S. Großmann, “The resource sharing system: dynamic federate mapping for HLA-based distributed simulation”, Proc. of the 15-th workshop on Parallel and distributed simulation (PADS’01), May 2001, Lake Arrowhead

[25] M. Myjak, S. Sharp, W. Shu, W. Wei, J. Riehl, D. Berkley, P. Nguyen, S. Camplin, and M. Roche, “Implementing object transfer in HLA”, Proc. 5-th Simulation Interoperability Workshop (SIW’99), Orlando, Florida, March 1999

[26] V. Naoumov and T. Gross, "Simulation of Large Ad Hoc Networks" in proc. MSWiM 2003, San Diego, CA, Sept. 2003

[27] K. Perumalla, R.M. Fujimoto, and A. Ogielsky, “TeD - A language for modeling telecommunications networks”, Performance Evaluation Review 25(4), 1998

[28] D.M. Rao, and P.A. Wilsey, “An Ultra-large Scale Simulation Framework”, Proc. of MASCOTS '99, Oct. 1999

[29] D.M. Rao, and P.A. Wilsey, “An object oriented framework for parallel simulation of ultra-large communication networks”, proc. 3-rd Inter.l symposium on computing and object oriented parallel environments, Nov. 1999

[30] D.M. Rao, and P.A. Wilsey, “Parallel Co-simulation of Conventional and Active Networks”, Proc. of MASCOTS’00, August 2000

[31] G.F. Riley, R.M. Fujimoto, M.H. Ammar, “A generic framework for parallelization of network simulations”, Proc. of MASCOTS'99, College Park, MD, October 1999

[32] G.F. Riley, M.F. Ammar, R.M. Fujimoto, K. Perumalla, and D. Xu, “Distributed Network Simulations using the Dynamic Simulation Backplane”, MASCOTS' 01, Aug. 2001

[33] G.F. Riley, and M.H. Ammar, “Simulating Large Networks How Big is Big Enough?”, Proc. of First Intern.l Conference on Grand Challenges for Modeling and Simulation, Jan. 2002

[34] J. Short, R. Bagrodia, and L. Kleinrock, “Mobile wireless network system simulation”, Wireless Networks 1, August 1995

[35] T.K. Som, and R.G. Sargent, “Model structure and load balancing in optimistic parallel discrete event simulation”, Proc. of the 14-th workshop on Parallel and distributed simulation, May 2000, Bologna

[36] K. Tang, M. Correa, and M. Gerla, “Effects of Ad Hoc MAC Layer Medium Access Mechanisms Under TCP”, ACM/Kluwer Mobile Networks and Applications, 2001

[37] UCB/LNBL/VINT The NS2 network simulator, http://www.isi.edu/nsnam/ns/

[38] A. Varga, OMNET++ in "Software Tools for Networking", IEEE Network Interactive. July 2002, Vol.16 No.4

[39] V-Y Vee, and W-J Hsu, “Locality-preserving load-balancing mechanisms for synchronous simulations on shared-memory multiprocessors”, Proc. of 14-th workshop on Parallel and distr. simulation, May 2000, Bologna, Italy, p.131-138

[40] X. Zeng, R. Bagrodia, and M. Gerla, “GloMoSim: A library for parallel simulation of large-scale wireless networks”, Proc. of Workshop of Parallel and Distributed Simulation (PADS’98), Banff, Alberta, Canada, May 1998