A New Adaptive Middleware for Parallel and Distributed Simulation of Dynamically Interacting Systems

Luciano Bononi, Michele Bracuto, Gabriele D'Angelo and Lorenzo Donatiello
Dipartimento di Scienze dell'Informazione, Università degli Studi di Bologna,
Mura Anteo Zamboni 7, 40126, Bologna, Italy
{bononi, bracuto, gdangelo, donat}@cs.unibo.it

Abstract

In this work we define and test a new framework obtained as the integration of two recently developed middlewares defined to support the parallel and distributed simulation of large scale, complex and dynamically interacting system models (like wireless and mobile network systems). In a distributed simulation of highly interacting system models, the main bottleneck may become the communication and synchronization required to maintain the causality constrains between distributed model components. We designed and implemented the ARTÌS middleware as a new framework incorporating a set of features that allow an adaptive optimization of the communication layer management in a distributed simulation scenario. ARTÌS has been integrated with GAIA, a dynamic mechanism for the runtime management and adaptive allocation of model entities in a distributed simulation. By adopting a runtime evaluation of causal bindings between model entities GAIA adapts the dynamic and time-persistent causal effects of model interactions to dynamic migration of model entities. Preliminary results demonstrate that the combined effect of ARTÌS management and GAIA heuristics leads to a significant reduction in the communication and synchronization overheads between the physical execution units. Simulation performance enhancements have been obtained also in worst-case modelling assumptions and simulation scenarios.

1. Introduction

In recent years, the research for tools and methodologies for modeling and simulation of large-scale and complex systems has obtained a great interest. Examples of models challenging currently available simulation systems and tools range from large-scale wireless systems, like cellular, mobile ad hoc and sensor networks, up to biology-inspired models and molecular systems, elementary particles physics and cosmology systems [17, 19, 20, 24, 26, 31, 35, 36, 37, 39]. The simulation-based investigation of complex systems is widely adopted and it is often preferred, in practice, to the mathematical intractability and critical model-complexity of alternative numerical and analytical resolution methods [17, 25, 30, 31, 35]. Simulation models currently considered interesting for the analysis may include a potentially huge number of simulated objects. The simulation of many objects may require a relevant computation time (e.g. due to the implementation of complex behaviors, with dense state changes). Detailed and complex simulated objects (i.e. model components) may require complex and large data structures implementing the model state. Large scale and complex simulation models may be 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 [25, 30, 31].

Many practical experiences have demonstrated that the memory bottleneck reduction, and the speed-up in the simulation of complex systems, can be achieved by using parallel and distributed models and execution architectures, i.e. a Parallel Discrete Event Simulation (PDES) approach [1, 6, 12, 13, 18, 19, 20, 28, 29, 31, 36, 39]. More recently, the distributed simulation community contributed in the definition of a new standard, named IEEE 1516 "Standard for parallel and distributed modeling and simulation" [16]. The new standard defines rules and interfaces allowing for heterogeneous components' interoperability in parallel and distributed simulations. Model components (formally known as federates) are executed as Logical Processes (LPs). Federates' execution is supported by standard management APIs for the communication and synchronization tasks, implemented by a runtime middleware (RTI). The High Level Architecture (HLA) has currently become a synonymous for the standard rules and services to be considered as the basis for the implementation of distributed simulations and the RunTime (RTI) simulation kernel [7, 11, 16]. In order to exploit the maximum level of computation parallelism, many research activities dealt with dynamic balancing of logical processes' executions (both cpu-loads and virtual time-advancing speeds) by trading-off communication, synchronization and speedup, both in optimistic and conservative approaches [5, 8, 10, 14, 32, 38].
The distributed federates interact and synchronize via event-message notifications (i.e. basically message

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

passing communication). Unfortunately, the need for distributed model-components communication and synchronization services may require massive interprocess communication to make the distributed simulation evolving like in the sequential counterpart. 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, and solutions to reduce the cost of communication must be addressed by the research in this field.

Many approaches have been investigated in order to reduce the overhead effects of distributed synchronization and communication in both optimistic and conservative distributed simulations. Solutions have been proposed, based on both model aggregation and on communication filtering, and also by trading off model accuracy and computation load balancing issues, respectively [15, 27]. Basically, the approaches defined in [2, 6, 10, 11, 21, 32] rely on the reduction of communication obtained when the update of an event- or state-information (e.g. event and/or anti-message) is not flooded to the whole system, but it is only propagated to the subset of causally dependent components. This is the basis of publishing/subscribing mechanisms for sharing state-information and event-notifications between causally dependent components [7, 11, 28]. In spite of the previously mentioned approaches for communication reduction, the efficient implementation of interprocess communication remains a primary background issue, to contrast the possible communication bottleneck of parallel and distributed simulations. The way interprocess communication can be sustained in distributed systems would depend mainly on the execution units and communication support, that is, on the simulation system resources, architectures and characteristics.

Recently proposed and implemented middleware solutions based on the IEEE 1516 Standard for distributed simulation and the High Level Architecture (HLA) [7, 16] have shown that the parallel and distributed simulation of massive and complex systems can result in relevant overheads. Overheads are due to the complex and full management of a wide set of runtime services and to the latency due to distributed communication bottlenecks. Specifically, the implementation of the interprocess communication services has been implemented in sub-optimal way, without considering the heterogeneity of the simulation execution platforms [3, 9].

To this end, we designed a new, parallel and distributed simulation middleware named Advanced RTI System (ARTÌS). The aim of the ARTÌS middleware is to support parallel and distributed simulations of complex systems, based on a minimum set of middleware services. The ARTÌS design is oriented to support the model components' heterogeneity, distribution and reuse, and to increase the simulation performances, scalability and speedup, in parallel and distributed simulation scenarios [4]. Another design issue of the ARTÌS framework is the dynamic adaptation of the interprocess communication layer to the heterogeneous communication support offered by possibly different simulation-execution units [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].

In addition, in this work the ARTÌS middleware has been composed with another distributed mechanism, named Generic Adaptive Interaction Architecture (GAIA). GAIA implements a simple model components' migration mechanism that can be adapted on the top of HLA-based distributed simulations [3]. The HLA standard and existing Runtime Infrastructures (RTI) do not define component migration facilities, even if preliminary research activity is made on this topic [22, 23]. For this reason we realized a prototype migration framework, and a heuristic migration policy, whose aim is to dynamically partition and allocate the interacting model components over many LPs, respectively executed over a set of multiple, distributed execution units. The composition of ARTÌS and GAIA would realize a complete prototype framework for parallel and distributed simulation, characterized by an adaptive, tuneable mechanism able to adapt and react to dynamic systems' behavior under the communication-reduction viewpoint. In this work the prototype implementation of the ARTÌS and GAIA mechanisms is outlined and preliminary results of a set of simulation tests for dynamically interacting model components are presented.

The paper structure is the following: in section 2 we outline some concepts about the distributed simulation of dynamic models, specifically, the wireless ad hoc networks; in section 3 the key issues for the ARTÌS and GAIA framework implementation and the proposed migration heuristics are defined; in section 4 a prototype wireless system's model and a preliminary set of simulation results are presented; in section 5 we summarize our conclusions and future work.

2. Distributed simulation of dynamic models

We define a *dynamic system* as a system where the interactions (i.e. the causal effects of events) are dynamically subject to fast changes driven by the system (and model) evolution over the simulated time. Given this general definition, a wireless network can be an example of a highly dynamic system.

To realize a correct evolution under the event-causality viewpoint, every model components' interaction should be notified as an event-message to all the causally dependent model components, 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 sustained in distributed systems would depend mainly on the execution units and on the communication support, that is, 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 and high communication latencies over local and wide area network communication services. It is self evident how 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 highly dynamic systems any optimal static clustering and allocation, based on the current component-interaction scheme, will become immediately suboptimal, due to the dynamics of the model interactions. The approach used in currently available implementations is to consider the model component interactions, by adapting the event message distribution accordingly. No background optimization is based on the heterogeneity of available communication infrastructure characteristics.

In presence of a dynamic system, 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 interaction scheme would be maintained for a significant time (i.e. time-locality).

2.1 A case study of a dynamic model

In the following, as an example of a dynamically variable system, we focus on a wireless multi-hop Mobile Ad Hoc Network (MANET) [17, 35]. Simulation models for wireless systems incarnate the assumptions that motivated our design. The number of simulated hosts in our expectations can reach high values, requiring the simulation of massively populated scenarios. Topology changes due to simulated hosts' mobility map on causality effects in the "areas of influence" of each mobile device, resulting in dynamically shaped causality-domains and component interaction schemes. Given two or more neighbor-hosts sharing the wireless medium, the causal effect of signal interference could result in a chain of local-state events up to the high protocols' layers [35]. In our approach, we define a model entity as the data structure defined to model a Simulated Mobile Host (SMH). A certain degree of time-locality of local communication can be considered an acceptable assumption in many wireless system models, depending on the communication load and the mobility model assumptions.

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 [17, 35]. Under the modeling and simulation viewpoint, wireless systems can be considered highly dynamic systems: if a SMH changes its position, it will eventually interact with a new community of neighbor hosts. The system dynamics can be influenced by motion model and speed, and also by the SMHs density.

Our testbed consists of a distributed discrete event simulation of model components (i.e. logical processes) executed over a set of physical execution units (PEUs), connected by a physical LAN network. 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 (RTI) 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 obtain the advantage of closing the causality effect of modeled communication within the PEU where the interacting LPs (and respective SMHs) are executed. In addition, clustered interacting SMHs would limit interactions with the management layers of the RTI, 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 fast simulation.

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 explicitly 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 optimal distribution ineffective 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 migrating the SMHs between LPs. In this way, the modeler is alleviated 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

The HLA implementation criticisms [38, 3, 4, 9] and the lack of Open Source RTIs are the main motivations behind the design and implementation of ARTÌS (Advanced RTI System). 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. 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 our target models of interest [3], which may result in frequent rollbacks. In ARTÌS, many design optimizations have been applied to obtain adequate protocols for synchronization and communication in Local Area Network (LAN) or Shared Memory (SHM) multiprocessor architectures. 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.

3.1. The Generic Adaptive Interaction Architecture (GAIA)

The PDES simulator built to obtain an experimental evidence of our proposal is based on a distributed architecture made by a set of logical processes glued together by the ARTÌS middleware. In [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 in our previous work, 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 overheads.

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 a mobile wireless system.

3.1.1 The heuristic migration-policy definition

The dynamic migration of simulated hosts is not free of costs: some analytical or heuristic metrics are required, to be evaluated at runtime, to define "if and where" it would be profitable to migrate a SMH. The state size of a SMH and the amount of "time-locality" of the causal dependency between neighbor hosts, are the most relevant parameters influencing the migration policy. Specifically, the policy depends on the motion models, the interaction rate between SMHs, and the overall load balancing between the PEUs. By focusing on the network communication-reduction viewpoint, it would be optimal to allocate every object on a single PEU, by running the distributed simulation over a single PEU. Obviously, this is not the intended purpose of the GAIA mechanism: the external communication-reduction needs a trade-off with effective load-balancing of the parallel executions. The optimal policy would require to dynamically partition the sets of the most frequently interacting SMHs model components, by allocating them over the PEUs in a perfect load-balanced way. This problem could be NP-hard because, depending on the model assumptions, it may be defined as a variation of the multiple knapsack problem. Anyway, we implemented a combination of two low-cost heuristic schemes, that adaptively converge to a near-to-optimal solution, under the system assumptions considered in the implementation. The heuristic migration rules are quite simple and have been improved with respect to the early design of our previous work [3].

Let a tagged SMH(j) be executed on the *i-th* PEU. Let us define *Rj_e=Me/Mi* as the ratio of the *Me* "external" messages sent to the *e-th* PEU, with respect to the number (*Mi)* of "local" messages sent within the local (*i-th*) PEU. Every SMH(j) evaluates the defined ratio *Rj_e* for every foreign *e-th* PEU. If the maximum ratio obtained is greater than a global threshold-value *K*, then the corresponding PEU is chosen as the candidate destination for the SMH(j) migration in the next timestep. No migration is performed, otherwise. Upon arrival on a new PEU, every SMH resets its message counters (*Mx*). The value of *K* is a simple tuning parameter that can be used to control the rate of migrations and the threshold of external communication required in order to balance the migration overhead. The proposed algorithm leads to good performances, but the information collected by every SMH during the whole simulation run must be upper bounded. The inclusion of old aged events information can bias the migration heuristic estimates. Current implementation is based on periodic resets of the SMH estimates (every fixed amount of time-steps). A reset is performed also after every SMH migration. A Sliding Window scheme for recent event messages will be investigated as future work.

3.2. The heuristic load-balancing policy definition

The steady state behavior of the migration heuristic in isolation would lead to the asymptotic clustering of all the SMHs over a restricted set of the available execution units. This is because the adaptive effect of migrations is focused on the reduction of "external" communication overheads. The migration heuristic must be composed with a load balancing policy, and the heuristics' tradeoffs should be optimized in orthogonal way. The load balancing strategy implemented by the GAIA middleware defined in [3] has been redesigned in a straight-forward and approximate way: SMHs migration towards/from a tagged PEU is possible whenever a perfect load balancing can be achieved. The simplifying assumptions supporting the load balancing scheme are: only one LP is implemented over a PEU, and every PEU must execute the same number of SMHs instances. This translates on the assumption that every PEU implements a uniform fraction of the total number of SMHs in the system. Previous experience with GAIA shown that dynamic fluctuations in the balancing strategy could lead to computational asymmetry for PEUs, affecting the simulation speedup. A crowded LP can become a synchronization bottleneck for the system. More specifically, the load balancing mechanism governs the migration heuristic, by allowing only balanced migrations between every pair of LPs. A three-phases migration procedure is the result of our implementation: in the first round every LP must claim the number of candidate migrations and their destinations. In the second round the balancing condition is evaluated, and in the third step all the migrations that match the load balancing rules are performed.

4. Model definition and experimental results

4.1. Wireless system's model definition

Now we illustrate the key concepts of our target wireless system and model definition. We assume a high number of simulated mobile hosts (SMHs), each one following a Random Mobility Motion model (RMM). This motion model is far from being real, 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 mechanism definition, our feeling is that any other widely used motion model, like any restricted, correlated or Group Mobility model, would result better than the adopted RMM model, for any migration heuristic. In the following, the RMM model is defined. SMHs swings between mobile and static epochs. At the beginning of each epoch, every SMH decides to stay or to change its mobile or static state, 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 by every SMH. Sometimes we considered two motion sub-models related to the motion speed: (slow-mobility) S-RMM and (fast-mobility) F-RMM. The F-RMM model is characterized by high speeds (25 spaceunits/timestep), and S-RMM is based on lower speeds (10 spaceunits/timestep). To stress the migration scheme, we have also used an extreme sub-model with very high speed (100 spaceunits/timestep).

Space is modeled as a torus-shaped 2-D grid-topology, 10.000x10.000 spaceunits, populated by a constant number of SMHs. The torus space topology, indeed unrealistic, is commonly used by modelers to prevent non-uniform SMHs' concentration in any area. 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. In our analysis we have investigated the impact of different ping-message sizes on the migration mechanism effect. We plan to extend this model with the real implementation of message flows, routing protocols and applications as a future work.

4.2. Experimental results

The set of experiments and the analysis shown in this section is similar to the analysis reported in [3] for the GAIA over HLA RTI system. Anyway, in this work, both the GAIA and the ARTÌS frameworks realized a completely different tool for simulation than the preliminary tool analyzed in [3]. The GAIA middleware has been completely reimplemented, and both the migration and the load-balancing heuristics have been completely redesigned. Moreover, the composition of GAIA with ARTÌS results in lower management overheads and greater speedup than the framework architecture described in [3].

The experiments were executed over a variable set of M PEUs each one equipped by Dual Xeon Pentium IV 2800 Mhz, 3 GB RAM, connected by a Fast Ethernet (100 Mb/s) LAN. We performed multiple runs, and the confidence intervals obtained with a 95% confidence level are lower than 5% the average value of the performance index shown.

In the following we define as "static" or "dynamic" a distributed simulation with the migration heuristic turned OFF or ON, respectively. All the performed experiments were started with a pseudo-random, uniform distribution of a variable number of SMHs (3.000 up to 9.000) over a grid topology (10.000 x 10.000 spaceunits). 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 our migration mechanism. Moreover, the random distribution would be asymptotically obtained in a "static" simulation, 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 mechanisms.

4.2.1 Initial and runtime distribution. A graphical representation of the dynamic SMH allocation between PEUs is shown in figure 1 and figure 2.

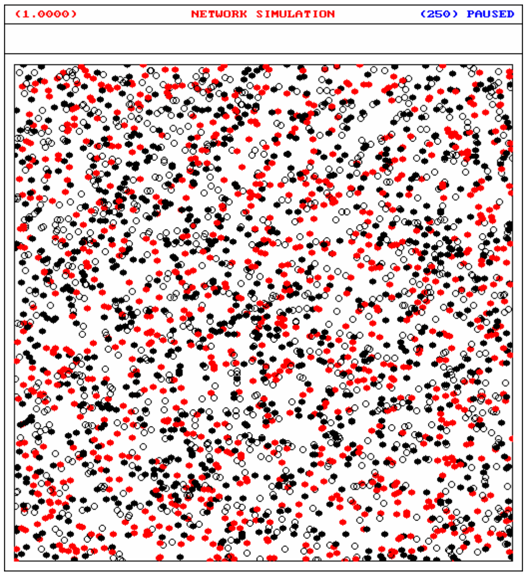

Figure 1. 3000 SMHs, initial random SMHs distribution, and steady state random distribution in "static" simulation, over PEU1 (black), PEU2 (white) and PEU3 (red or gray)

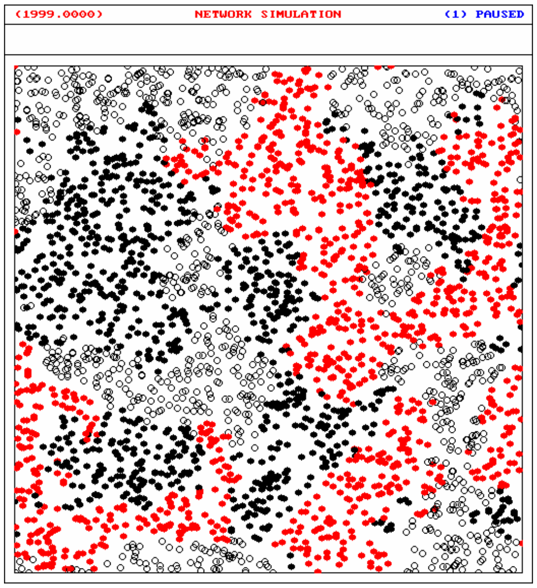

Figure 2. 3000 SMHs, steady state SMH distribution in "dynamic" simulation, over PEU1 (black), PEU2 (white) and PEU3 (red or gray)

Both figures show a snapshot of the random distribution of 3000 SMHs distributed on the 2D grid space. Every dot in the figures represents the current position of a SMH in the simulated area. Dot colors illustrate which PEU is executing the SMH. Three PEUs have been considered in this phase. Black dots refer to approx. 1000 SMHs allocated on PEU1, white dots refer to approx. 1000 SMHs initially allocated on PEU2 and red (gray) dots refers to remaining 1000 SMHs allocated over PEU3. Every SMH transmits a sequence of broadcast messages to all other SMHs located within its transmission range. A sample of the initial (and runtime) random SMH allocation over three PEUs in a static distributed simulation (that is, no migration) will appear as in figure 1. Under the system model assumptions, every SMH interacting with a subset of neighbor SMHs would have on the average 33% of its neighbors belonging to every available PEUs. After few time-steps from the initial random allocation of SMHs, the steady-state allocation obtained by the migration mechanism in a dynamic simulation appears like in figure 2. It results clear the clustering of highly interacting SMHs, obtained and maintained at the steady state, independently from the initial node allocation and despite the high SMHs' mobility.

4.2.2. The local communication ratio (LCR). The simulations performed mainly focused on the evaluation of the communication cost needed to implement the model interactions (i.e. event-messages) between SMHs. We define as a "local communication" (LC) a shared memory communication between SMHs clustered on the same PEU. On the other hand, an "external" communication (EC) is a message involving a physical network communication between different PEUs. For every PEU, we collected results regarding the *local communication ratio LCR=LC/(LC+EC)*. LCR results have been collected and analyzed with the heuristic migration policy respectively ON and OFF (i.e. with a static allocation), with respect to the SMH density and with respect to the value of the migration control parameter *K*. The *LCR* index is not related to the size of messages and describes how much the causality effects are closed inside each LP by adopting the migration mechanism. This index is not relevant about the amount of speedup obtained, because it does not describe the communication overhead for the objects' migration and data distribution management. Despite the mobility model and the model dynamics, the LCR demonstrates that a given percentage of messages required to perform the simulation runs can be transformed from ECs to LCs. In figure 3 we show the transient percentage of local communication *(LCR)* as a function of the modeled SMH density and the SMH speed, in static and dynamic simulations. SMHs are initially distributed randomly over the set of PEUs, like in figure 1. When the migration is on (K=3), the adaptive runtime re-allocation of SMHs increases the percentage of local communication, almost independently by the SMH density (3000 up to 9000

SMH in the area) and by the average SMH speed (Fast-RMM and Slow-RMM).

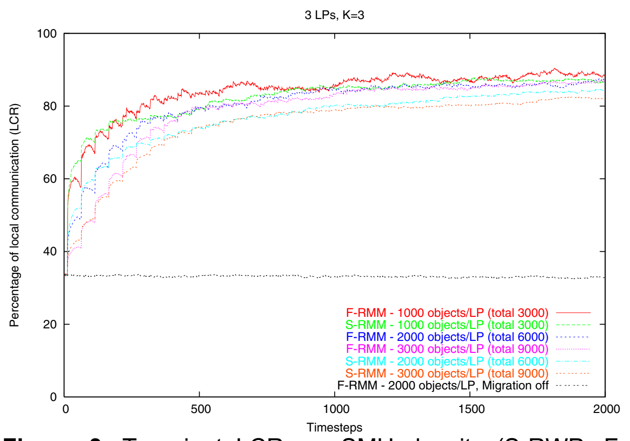

Figure 3. Transient LCR vs. SMH density (S-RWP, F-RWP)

A static simulation (migration off) always maintains the system on a flat level of local communication ratio (about 33%) as expected for this scenario. The steady state LCR for the dynamic simulation gives higher percentage of local communication than the static one (up to 88% for Fast-RMM, in figure 3). The higher the SMH speed, the higher the LCR convergence to the steady state value. The obtained values outperforms the results obtained in [3]: as an example, 61% vs. 88% LCR for the same scenario.

Figure 4 shows the transient LCR results by varying the K value defined to control the migration heuristic. As expected, low K values make the initial (and transient) re-allocation faster than high K values, and marginally affect the steady state LCR. These results show that the K parameter can control the convergence speed to the steady state, and high K values limit the number of migrations performed. The dynamic system converges to a steady-state *LCR* around 88% for Fast-RMM, and around 84% for Slow-RMM motion models (see figure 4).

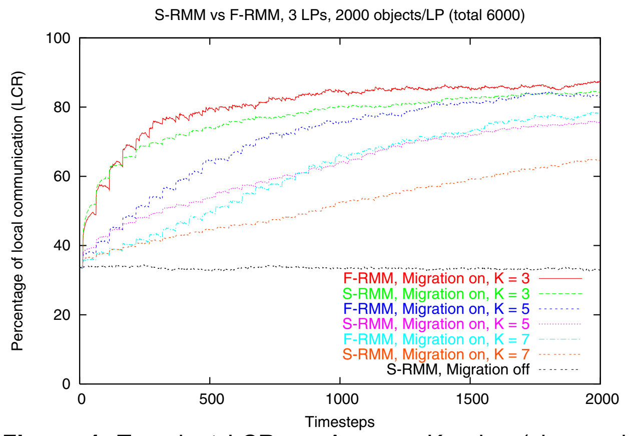

Figure 4. Transient LCR vs. Average K value (slow and fast RMM mobility)

The effect of the underlying load-balancing mechanism is transparent in these results. The differences between F-RMM and S-RMM is a clear indication of the different "time-locality" effect and persistence of interactions which is captured by the migration heuristics.

Figure 5 shows what happens if the migration mechanism is switched off at runtime (i.e. timestep 1000 on figure 5). The LCR ratio converges to the average value of a random, static allocation scheme. This convergence ratio is influenced by the motion model speed: a high SMH speed translates to fast convergence. This demonstrates the "time-locality" effect which is captured by the migration heuristic, and the fact that any initial, optimal, static allocation policy would not be adequate for this kind of dynamic models.

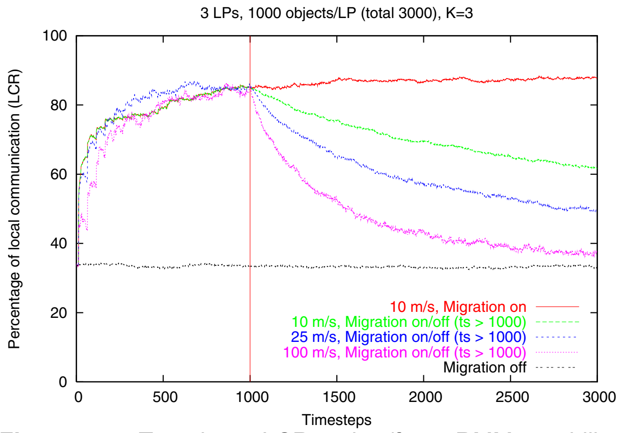

Figure 5. Transient LCR, slow/fast RMM mobility, migration mechanism turned OFF at timestep 1000

4.2.3. Execution-time analysis. In table 1 we show the wall-clock time required for simulating the initial 1000 timestep of simulated-time interval, by adopting different execution configurations. We are not describing the speedup index obtained by parallel and distributed simulations with respect to a monolithic sequential simulation. This choice is made since we defined our model as a worst case scenario with respect to the sequential/parallel speedup analysis (e.g. low local computation, frequent communications, low persistence of interactions due to random uncorrelated mobility). The asymptotic speedup of the proposed implementations, and scalability beyond 3 PEUs will be evaluated as a future work. What we are interested in, is the evaluation of the static and dynamic approaches for parallel and distributed simulation, to demonstrate that our migration-based mechanism could outperform a static approach, even in the worst-case scenario. As a worst-case scenario we mean a scenario where a high degree of uncorrelated mobility combined with frequent communication is performed. Both static SMHs scenarios, and limited-communications, would reduce the need for migrations, returning the static (migration-off) performance.

Before commenting the wall-clock time data shown in table 1, it is worth noting the simulated time of a simulation run is limited to only 1000 timesteps. This is a really short simulation run. Significant run-length for simulations would be of many thousands timesteps, depending on the convergence time and variance of simulation indices.

Table 1. Preliminary execution-time results (single run execution, 1000 timesteps)

M PEUs, **N** federates, **5000** SMHs (constant)	Migration	Wall Clock Time (s) 1000 ts
M = N = 1	Off	23 min, 38 sec
M = 1, N = 3	Off	20 min, 40 sec
M = 1, N = 3	On, K=3	18 min, 42 sec
M = N = 2	Off	18 min, 01 sec
M = N = 2	On, K=3	16 min, 03 sec
M = N = 3	Off	14 min, 30 sec
M = N = 3	On, K=3	12 min, 36 sec
M = N = 3	On, K=7	14 min, 17 sec

The wall-clock time indicated in Table 1 includes the initial re-allocation, and object distribution management overhead, which is characteristic of the migration mechanism, in the dynamic approach. Despite the initial migration overheads, the results show that a simulated time of only 1000 timeslots is sufficient for our implementation to recover all the initial and runtime migration overheads. Overheads are almost immediately balanced by the external messages' reduction. This is even more relevant given all the worst-case assumptions about the light local computation for each SMH. The results presented in Table 1 shows that the parallel and distributed simulation of the referenced model always outperforms both a monolithic simulation, and the static distributed simulation. By increasing the number of PEUs involved in the simulation we always obtain a reduction of the wall-clock time required for the initial timesteps. By simply activating the migration framework the wall clock time required by a dynamic distributed simulation reduces of 10%, 12% and 13% for 3 LPs on 1 PEU, 2 LPs on 2 PEUs, and 3 LPs on 3 PEUs, respectively. The gain indicated is relative to the static distributed simulation. Results presented in Table 1 have been obtained with the worst case modeling assumption that a ping message contains no payload (i.e. header only). We expect that the increase of the ping message would result in the amplification of the communication overheads, and in additional advantages of the dynamic simulation, which increases the LCR. Figure 6 illustrates the wall clock times obtained by increasing the payload size of the modeled ping messages in the simulations. Small payload sizes (up to 64 bits) result in a 10-15% speedup of the dynamic vs. static distributed simulation. By increasing the payload up to 1024 bits the speedup gain is greater than 120%.

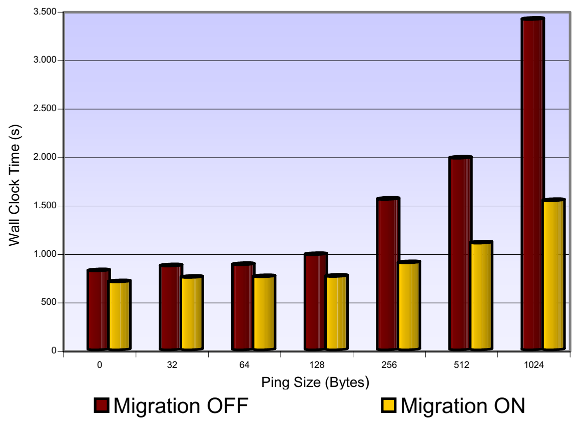

Figure 6. Wall Clock Time vs. Ping Packet Size

5. Conclusions and future work

In this work we defined and 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 and dynamically interacting 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. By adopting a runtime evaluation of causal bindings between model entities GAIA adapts the dynamic and time-persistent causal effects of model interactions to dynamic migration of model entities. We tested our adaptive framework and the migration and load-balancing heuristics in the testbed simulation of a prototype mobile wireless system, characterized by Simulated Mobile Hosts (SMHs). The results obtained demonstrate that the combined effect of ARTÌS management and GAIA heuristics leads to a significant reduction in the communication and synchronization overheads between the physical execution units. Simulation performance enhancements have been obtained also in worst-case modeling assumptions and simulation system scenarios.

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, wireless ad hoc and sensor networks, biology-inspired models and molecular systems, elementary particles physics and cosmology systems.

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

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

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

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

[10] 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

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

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

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

[14] 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

[15] 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

[16] 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

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

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

[19] 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

[20] 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

[21] B. Logan, and G. Theodoropoulos, "The Distributed Simulation of Multi-Agent Systems", Proc. of the *IEEE*, 2001

[22] 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

[23] 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

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

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

[26] 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

[27] D.M. Rao, and P.A. Wilsey, "Parallel Co-simulation of Conventional and Active Networks", Proc. of MASCOTS'00, August 2000

[28] 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

[29] 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

[30] 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

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

[32] 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

[33] B.K. Szymanski, Y. Liu and R. Gupta, "Parallel network simulation under distributed genesis", Proc. of the 17-th workshop on Parallel and distributed simulation, 2003, Washington DC, USA

[34] B.K. Szymanski, and Y. Liu, "Loosely-coordinated, distributed, packet-level simulation of large-scale networks", Proc. of the Winter Simulation Conference, December 2003

[35] 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

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

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

[38] 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

[39] 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