Proximity Detection in Distributed Simulation of Wireless Mobile Systems

Luciano Bononi Michele Bracuto Gabriele D'Angelo Lorenzo Donatiello

Dipartimento di Scienze dell'Informazione, Università degli Studi di Bologna
Mura Anteo Zamboni 7, 40127, Bologna, Italy

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

ABSTRACT

The distributed and the Grid Computing architectures for the simulation of massively populated wireless systems have recently been considered of interest, mainly for cost reasons. Solutions for generalized proximity detection for mobile objects is a relevant problem, with a big impact on the design and the implementation of parallel and distributed simulations of wireless mobile systems. In this paper, a set of solutions based on tailored data structures, new techniques and enhancements of the existing algorithms for generalized proximity detection are proposed and analyzed, to increase the efficiency of distributed simulations. The paper includes the analysis of computation complexity of the proposed solutions and the performance evaluation of a testbed distributed simulation of ad hoc network models. Recent works have shown that the performance of distributed simulation of dynamic complex systems could benefit from a runtime migration mechanism of model entities, which reduces the communication overheads. Such migration mechanisms may interfere with the generalized proximity detection implementations. The analysis performed in this paper illustrates the effects of many possible compositions of the proposed solutions, in a real testbed simulation framework.

Categories and Subject Descriptors

I.6.8 [**Simulation and Modeling**]: Types of Simulation – discrete event, distributed.

General Terms

Algorithms, Performance, Design, Experimentation.

Keywords

Proximity detection, wireless systems, distributed simulation, data distribution management.

1. INTRODUCTION

The computer simulation is a powerful technique to model, evaluate and predict the behavior of complex dynamic systems. Many systems are composed by a great number of entities, dynamically interacting in unpredictable way. The analysis approach based on mathematical modeling is often inadequate to express the fine details and complex interactions of such systems. The simulation can be considered a viable solution in these cases. A discrete event simulation mimics the behavior of a system model at discrete points in time. The interactions between simulated entities are represented by simulation events and the processing of such events causes the evolution of the simulation process [9].

A common problem of many simulations is to determine what entities are involved in the notification and processing of a given event. As an example, in a wireless simulation a new frame transmission causes the creation of one transmission event-message by the transmitter entity, that must be delivered in chronological (causal) order to the whole set of potential receiver entities. Due to the local broadcast nature of the wireless transmission, wireless signals decay with distance depending on propagation model assumptions [14]. For this reason, it makes sense to distribute transmission event messages only to the subset of neighbor hosts that will be reached by the transmission effects [12, 13]. Since hosts may be mobile, such subset is dynamic. A generalization of the subset definition problem is known in the literature as generalized proximity detection for moving objects [15]. Under the simulation viewpoint, the problem can be translated in the dynamic identification of the recipient entities of each event message. In a distributed simulation, this operation is usually executed by the event distribution management component. The conflict detection between moving objects is a more specific case of the general proximity detection algorithm [17]. A partially related field is given by physical particles' simulation: to determine the evolution of a single simulated entity it is necessary to take into account its interactions (forces) with all other entities in the system [1].

In the following, we will focus on distributed simulation of wireless models. Some work has been done in the past to address the proximity detection problem over single processor architectures [13], and multiprocessor architectures with shared memory [10,12,16]. On the other hand, distributed and Grid Computing architectures are gaining a lot of interest, mainly for cost reasons, even if in this case no shared memory would be available.

The paper structure is the following: section 2 introduces the motivations and basic concepts about distributed simulation; section 3 defines the basic assumptions about proximity detection and the related literature; section 4 introduces a set of proposals to enhance the proximity detection implementation in distributed simulations; section 5 introduces ARTÌS, that is, the distributed simulation framework adopted in the experimental evaluation of the proposed solutions, reported in section 6. Finally, section 7 illustrates our conclusions and future work.

2. THE DISTRIBUTED SIMULATION ARCHITECTURE

The traditional approach for computer simulation is monolithic, that is, a single CPU is running the whole simulation. In complex, detailed models and massive simulated scenarios such approach is less practical, both for the excessive time required to complete the simulation runs and for the limitations due to the memory constraints of one single processing unit [5]. A valid alternative is given by distributed simulation: a set of Physical Execution Units (PEU) is working together to run the simulation, whose implementation is split in many building blocks. A common execution architecture for distributed simulation is composed by a set of multiprocessors interconnected by a network (LAN, WAN or even Internet). By composing the PEUs architecture, aggregation of memory and computation resources gives advantages under the scalability viewpoint. If shared memory is unavailable, the distributed system has to pay some communication overheads to maintain a consistent and synchronous state. As an example, the latency of a LAN is orders of magnitude higher than the latency of a shared memory communication. In a distributed execution architecture, the task of dynamically determining the set of entities whose execution is interested by a new event can have a high cost. Therefore, data distribution techniques are required to propagate the relevant data, by reducing the communication overhead. In a distributed simulation, the communication cost can be so high that a concurrent replication of the same computation over many PEUs could be even more efficient than distributing and sharing the result of a single computation.

A distributed simulation is composed by a set of Logical Processes (LPs) executed over separated PEUs, and interacting together by adopting a message passing communication paradigm. Each LP manages the evolution of one or more model components. Obviously, the monolithic and the distributed execution of a given simulation model have to produce the same results. In the case of a distributed architecture, each CPU works at a different speed and the communication latency is usually unpredictable. To obtain correct results, the LPs can not proceed asynchronously at full speed: a synchronization mechanism is required to maintain the event causality. Mechanisms for distributed synchronization have been proposed by following alternative classes of solutions. In this work we assume that all the LPs will be synchronized under a time-stepped execution: the simulated time is divided in constant time steps and the simulation will proceed by executing events occurring step by step.

3. BACKGROUND AND RELATED WORK

Determining the proximity set of a wireless transmitter node can be seen as a special case of the N-body problem [1,14]. The N-body problem consists in determining the evolution of a set of N bodies each one interacting with the forces exerted from each body to any other body. The resulting force between two bodies is a function of the bodies' distance. The force becomes negligible at large distances. Possible algorithms to solve the N-body problem are the Burnes-Hut (BH), the Fast Multipole Method (FMM) and the Parallel Multipole Tree Algorithm (PMTA) [1]. All the algorithms are based on a octree data structure for 3-D modeling scenarios, and quadtree data structures in 2-D where the simulated scenario is divided in a set of cells. The BH algorithm is based on body-cell interactions. The FMM is based on cell-to-cell interactions. The PMTA is a hybrid of the BH and the FMM algorithms [1]. The BH algorithm has been applied to compute the interferences in wireless cellular simulations [14]. Given a set of wireless devices moving in a simulated scenario, at each timestep, the simulator has to determine the whole set of hosts affected by the ongoing transmissions. In a simplistic way this problem can be solved by computing the distance between each pair of nodes. This algorithm is usually referred to as "all-pairs" (brute force) and has a complexity of $O(n^2)$ per timestep, both in the average and in the worst case.

In the following section we will introduce enhancements and alternative approaches for attempting to reduce the cost of the all-pairs mechanism. Our work is based on some basic assumptions of the wireless networks and on specifically designed data structures. A given algorithm to partition the simulated entities allocated on a set of LPs (and related PEUs) of a distributed simulation execution may have effects on the cost of proximity detection functions, as it will be shown in section 5. A parallel proximity detection solution for moving objects has been considered in [15], based on the Distribution List (DL) algorithm. In the DL algorithm the simulated space is modeled as fuzzy grids and a set of lists is used to detect proximity between movers and sensors. Another grid-based approach is illustrated in [16]: the space is divided into a grid of cells linked to a list of the nodes belonging to each respective grid cell. In this approach, the computation is staged to allow the reuse of previously computed and reusable proximity detection results. The D-SPANNER presented in [10] is another approach based on kinetic sparse graph spanner that maintains a set of assertions about the state of the system, and exploits knowledge or predictions about the node mobility.

In 2000, the IEEE 1516 High Level Architecture (HLA) [8,11] standard for distributed simulation has been approved. The main goal of this standard was to increase interoperability and reuse of simulations. In the simulations compliant with the standard, the proximity detection is demanded to the Data Distribution Management (DDM) module [7]. The DDM is based on a publish/subscribe mechanism. Expressions of interest are explicitly used to keep track, to manage and to reduce the data transmissions to the essential data notifications. In HLA, the interest management can be realized based on regions and classes. The region-based approach is used to express interest for the events occurring within a subscribed area of the simulated-space. The class-based approach allows subscribing objects, attributes and classes under a hierarchical objects' structure. The IEEE 1516 standard defines rules and interfaces for distributed simulation, and does not impose any requirement about DDM and underlying proximity set management implementations.

4. PROPOSED PROXIMITY MANAGEMENT SOLUTIONS

The intensity of the wireless signals decays as a function of the distance, depending on the considered propagation model. By assuming that all the devices have homogeneous receiver sensitivity, and all transmissions are isotropic in a wireless system simulation, we can assume that transmissions have a homogeneous maximum horizon distance limit for detection. In other words, hosts out of a defined range (r) from the transmitter are not subject to any significant transmission effect. Some enhancements aiming to reduce the computation cost of the all-pairs algorithm will be illustrated in the following, based on previous assumptions. The first two enhancements we propose are based on the horizon distance assumption. The third proposal is based on the commutative property of the distance operator. The computation complexity of the proposed methods will be informally described. In section 6, performance evaluation results of a wireless scenario will confirm the enhancements obtained.

In the following we assume a simulation model composed of a high number of Simulated Mobile Hosts (SMH), each one following a Random Mobility Motion model (RMM) (see section 6). The simulated space is modeled as a torus-shaped 2-D topology and it is populated by a fixed number of SMHs. In general the applied mechanisms could be extended to 3-D topology models. In the following, we will refer to space-units and time-units to generalize our analysis.

4.1 External and internal squares

Defining n as the number of simulated nodes (SMHs), the all-pairs algorithm computes $n(n-1)$ Euclidean distance values between nodes coordinates. In a 2D plane, the Euclidean distance between two points is given by

$$d(x,y) = |x-y| = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2} \qquad (1)$$

By exploiting the "homogeneous limited horizon distance" assumption of wireless transmissions, it is possible to reduce the effective cost of this algorithm. Given a static horizon range (r), it is possible to define two squares around each transmitter SMH, both centered on the SMH coordinates: the size of the external square side is defined as $2r$ space-units and it defines the maximum area where SMH transmission could be sensed. Hosts allocated outside of the external square can be assumed to be out of the reception range of the central SMH, and can be discarded without any further evaluation. The internal square is inscribed in the circle with radius r (the transmission range), and its side size is $r\sqrt{2}$ (Figure 1). The transmission events originated by the SMH can be delivered without any further examination to other SMHs within the inner square. The space between the external and the internal squares represents the area of uncertainty (the dashed zone in Figure 1): all the SMH in this area can be inside or outside the transmission range. In this case, to determine the effect of a transmission, the distance has to be calculated SMH-by-SMH for hosts in the dashed area, and the results compared to r value.

By assuming a 2-D space, in the worst case four comparisons are necessary to determine if a point falls within a square. In most cases it is sufficient to check the external square to discriminate an event: hence, in most cases, we can reduce the number of required operations and their complexity.

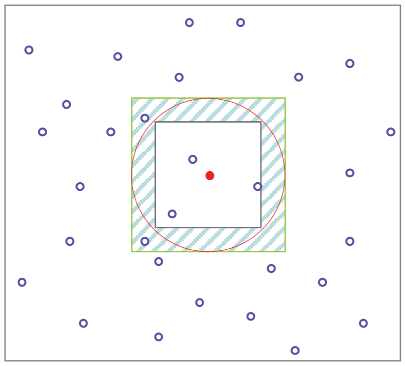

Figure 1. The communication range of each SMH can be bounded by an external and an internal square.

With respect of the classical all-pairs algorithm, the squares method still requires $n(n-1)$ comparisons between points but drastically reduces the number (C_d) of costly Euclidean distances to compute. The area of the external square is given by $A_{ext} = (2r)^2 = 4r^2$, the area of the internal square is $A_{int} = (r\sqrt{2})^2 = 2r^2$. By assuming $\rho = n/spaceunit^2$ as the homogeneous distribution of SMHs:

$$C_d = \left(4r^2 - 2r^2\right)\rho = 2r^2\rho \qquad (2)$$

C_d defines the average number of Euclidean distances to compute per time-step for each SMH in the simulation.

4.2 Grid-based data structure

In this solution, an overlay grid divides the simulated space in cells. Each square of the grid (or cell) represents only a small portion of the simulated scenario and has a fixed size. This approach requires to order the SMHs of the grid structure in a way that allows to reduce the computational load. The data structure implementing the overlay grid has to efficiently support the insertion and deletion of entries, to manage the SMHs mobility. In our implementation the overlay grid is composed by a matrix of separate chaining hash tables (Figure 2). If the simulated area covered by a cell is empty then also the relative hash will be empty (and not allocated), otherwise the hash table will contain all the SMHs that are located within the grid cell. A data structure based on a matrix of lists would be unable to efficiently manage the dynamic evolution of the simulated scenario, with respect to a matrix of hash tables.

The grid structure is useful to easily determine the neighbor set of each SMH: no need to inspect the whole simulated space. Only the grid squares whose area has non zero intersection with the transmission range circle have to be checked. In this way the number of checks can be drastically reduced, but the data structure management introduces some overhead.

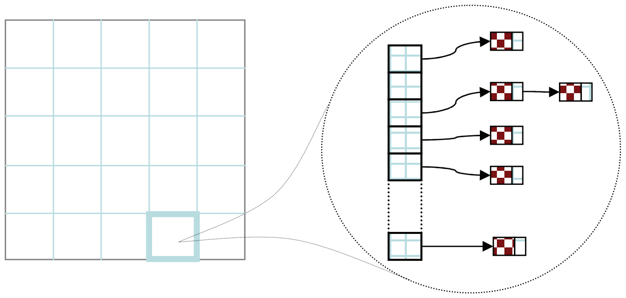

Figure 2. The overlay grid is composed by a matrix of separate chaining hash tables.

The grid-based mechanism works as a filter to reduce the number of checks, but it is worth noting that false positives can happen in cells overlapped by the transmission range circle (Figure 3). The reason is simple: the area covered by the grid squares is always equal or bigger than the transmission circle.

In general, the proposed mechanisms can be combined to further reduce the number of Euclidean distances to calculate. As an example, the filtering given by current grid-based solution, can be followed by the internal and external squares filtering. Finally, only the nodes surviving the filtering mechanisms have to be checked by computing the Euclidean distance with formula (1).

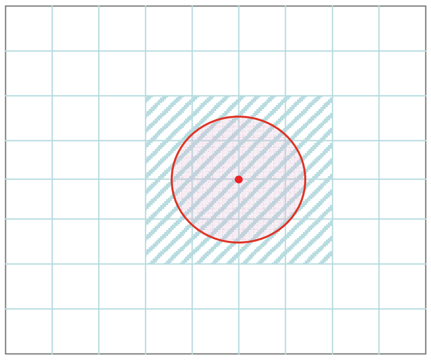

Figure 3. A set of cells is overlapped by the transmission range of each SMH. The cells at the borders can generate false positives.

It is quite obvious that the performance of the grid-based approach is drastically influenced by many factors, including the cell size. Many small cells would be able to reduce the number of false positives at the cost of an increased overhead for the data structure management. Conversely, a small number of big cells would reduce the management cost but would originate a high number of false positives. As it will be shown in section 6, assuming a uniform distribution of SMHs, the density (ρ) becomes a key factor to determine the mechanism performance. Defining g as the measure of the cell side, we have that each SMH has to check at worst $\left(\left\lceil 2r/g \right\rceil\right)^2$ grid cells. The following formula for C_d defines the number of Euclidean distances to compute per time-step for each transmitting SMH:

$$C_d = \left(\left\lceil \frac{2r}{g} \right\rceil\right)^2 g^2 \rho \quad . \tag{3}$$

The grid data structure has to be updated to reflect the SMHs mobility. Hence the cost of this approach also depends on the mobility model and on the cost to update the grid data structure.

4.3 Results caching

The Euclidean distance (as defined in section 4.1) is a commutative binary operator: given two points x and y we have that $d(x,y) = d(y,x)$. The commutative property can be used to reduce the number of operations needed for the all-pairs algorithm. In a centralized simulation-execution architecture it would be possible to reduce the number of Euclidean distance operations to $n(n-1)/2$. In both cases, the computation complexity is $O(n^2)$ as the classical all-pairs approach, but in the latter case, the constant factor hidden in the asymptotic notation is quite different. Again, this enhancement can be combined with the previously ones introduced, to obtain a multi-staged filtering and incremental computation reduction.

On the other hand, in a distributed simulation, each LP manages only a subset of the SMHs and a global knowledge of the simulated environment is missing. Moreover, in a distributed architecture, the communications needed to construct a global knowledge are costly and have to be controlled: each LP has a partial vision of the whole system and only the necessary information are propagated.

Assuming a homogeneous distribution of SMH entities over LPs, we have $\#SMH = n/\#LP$ allocated in each LP, where n is the total number of simulated mobile hosts and $\#LP$ is the number of logical processes in the simulation. A prerequisite to apply the commutative optimization is that if in a pair (SMH_1, SMH_2) both simulated nodes are allocated in the same LP, it would be possible to reuse the results of the distance computation, since the result can be already cached and available within the same LP. If the SMHs are allocated on different LPs then the distance value has to be computed by both the LPs (or transmitted between the two different LPs).

In the analyzed environment the probability that two SMHs are in the same LP is given by

$$P(x,y) = \left(\frac{\#SMH}{n}\right)\left(\frac{\#SMH-1}{n}\right) \tag{4}$$

Since $\#SMH$ is inversely proportional to the number of LPs in the system, in complex simulations we obtain that it is unlikely to apply the commutative optimization, because the simulated entities would be partitioned over a large set of LPs.

In the following section, we will show that a model entity migration-based mechanism can reduce the communication overhead, can support computation load balancing and increase the positive effects of the results caching.

5. THE ARTIS MIDDLEWARE

The Advanced RTI System (ARTÌS) is a middleware for Parallel and Distributed Simulation (PADS) supporting massively populated models [5]. The design of the middleware is inspired by the IEEE 1516 standard but new features have been introduced to improve scalability and simulation speed. The PADS execution speed is highly affected by the communication performance: the approach followed by ARTÌS is adaptive and exploits the LPs physical allocation. Specifically, a couple of LPs running on the same PEU will communicate using the low latency shared memory. In the case of LPs connected by LAN, WAN or Internet, they will communicate using the R-UDP/IP or TCP/IP protocols. The protocol choice is adaptive and based on the network performances. Other important features of the middleware are the multi-threading support, the implementation of data structures specifically tailored for the event-list management, the capability to detect and adapt to hyper-treading and multi-core CPUs, and the support for the simulation cloning and concurrent replication of independent runs [4].

In [6] it has been shown that the performances of a distributed simulation can be increased by introducing the migration of the simulated entities. A migration based middleware can adaptively optimize the simulation execution by reallocating the simulated entities over the LPs. The dynamic reallocation can reduce the communication overhead and improves the computation load balancing. This translates into a reduction of the Wall-Clock Time (WCT) needed to complete the simulation runs. The Generic Adaptive Interaction Architecture (GAIA) is a migration based framework integrated in ARTÌS. The basic task of GAIA is to check the communication pattern of each SMH during all the simulation execution. A set of heuristics evaluates the communication pattern and trigger the SMH reallocation to reduce the communication costs [2] and to improve the load-balancing of the execution architecture. GAIA clusters the highly interacting SMHs within the same LP, reducing costly inter-LP communication and increasing the rate of low cost intra-LP communications. The cost of migrating the simulated entities is a key factor to be evaluated in the migration heuristics. An analytic evaluation of this cost is impossible due to the network heterogeneity and the unpredictable behavior of the simulated system. Furthermore the load balancing between the PEUs is a strict requirement for the distributed simulation. Clustering all the SMHs in the same LP would be optimal to reduce the communication overhead, but would led to a monolithic simulation and consequently to a worst case load-balancing [3].

The benefits of the migration based approach in the simulation of wireless models have been evaluated in [2] and [3]. In this work we are interested in: i) enhancing the all-pairs algorithm for proximity detection mechanism, and ii) evaluating the effects of the entities migration on the performance of the proximity detection algorithms. To reuse the pre-computed distance values (results caching), both the two SMHs must be allocated on the same LP. According to formula (4), the probability that two randomly allocated SMHs are found on the same LP is inversely proportional to the number of LPs in the simulation. The migration mechanism works by clustering together the highly interacting SMHs. The direct consequence is that the migration mechanism will allocate together the SMHs that are neighbors in the simulated wireless topology. This would increase the probability that two SMHs in proximity are allocated in the same LP, resulting an effective pre-condition for the results caching.

6. TESTBED PERFORMANCE EVALUATION OF A WIRELESS MODEL

In the following the results collected in the distributed simulation of a testbed ad hoc network model will be presented to compare and test the effectiveness of the proposed solutions for proximity detection and entity migration of distributed simulation of wireless system models.

6.1 Simulation system and simulation model

All the experiments and the analysis results shown in this section are based on the distributed simulation of a wireless ad hoc network model, running on the ARTÌS simulation middleware. We performed multiple runs for each experiment, and the confidence intervals obtained with a 95% confidence level (not shown in the figures) are lower than 5% the average value of the performance index.

The experiments have been performed over 2 PEUs, each one equipped by Dual Xeon Pentium IV, 2800 MHz, 3 GB RAM, interconnected by a Fast-Ethernet (100 Mb/s) LAN. The distributed simulation is composed by 2 LPs each one statically allocated on a different PEU.

The model is composed by a high number (2000 up to 8000) of simulated wireless mobiles hosts (SMHs), each one following a Random Mobility Motion (RMM) with a maximum speed of 10 m/s. This mobility model is far from being real, but it is characterized by the completely unpredictable and uncorrelated mobility pattern of SMHs. 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. The simulated area is modeled as a torus-shaped bi-dimensional topology, 10.000x10.000 space-units. The torus area, indeed unrealistic, allows to simulate a closed system, populated by a constant number of SMHs. The torus space assumption is commonly used by modelers to prevent non-uniform SMHs concentration in any sub-area. The simulated space is flat and open, without obstacles. The modeled communication pattern between SMHs is a constant flow of ping messages (i.e. constant bit rate), transmitted by every SMH in broadcast to all neighbors within a wireless communication range of 250 space-units.

6.2 Experimental Results

In this section, the simulated model is evaluated with different proximity detection algorithms executed in background (that is, *all-pairs*, *all-pairs* with *squares* enhancement, and *grid*), caching strategies (on/off), and model entity migration (enabled/disabled). We evaluate the Wall-Clock-Time (WCT) necessary to complete a simulation run, as a function of the increasing number of SMHs.

The implementation of the algorithm based on *squares* does not require any additional data structure and only a few modifications in the source code. Figure 4 shows that the WCT obtained with the *squares* algorithm decreases with respect of the *all-pairs* algorithm, from 45% to 20%, when increasing the number of SMHs.

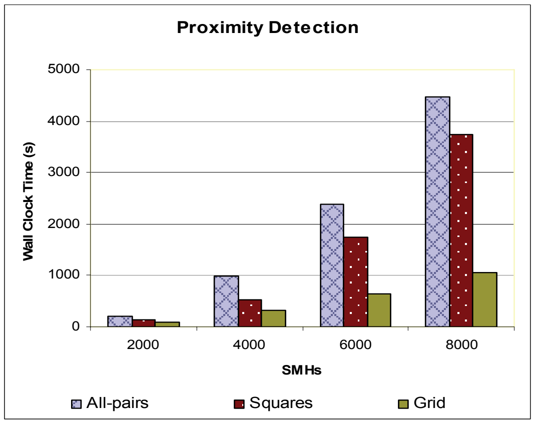

Figure 4. Proximity detection: evaluation of different algorithms and enhancements

The percentage performance gain decreases when the number of SMHs increases because the proximity detection algorithm has a reduced percentage computation load with respect of the whole simulation.

The *grid*-based approach has the best performances with respect of the *all-pairs* and the *squares* mechanisms. The *grid*-based approaches are really sensitive to the cell-size, the model density and other model peculiarities. Moreover, an implementation of the grid data structure for the simulation of wireless systems must cope with SMH model dynamics and their mobility. As an example, during a simulation run each SMH will roam a large number of cells, hence causing a lot of insertion and deletion operations in the grid data structure. This data structure is implemented as matrix of separate chaining hash tables, that is, efficient with respect to update costs, but has a significant memory requirement. In figure 5, we present the influence of the cell size (square side size) on the WCT of the simulated model: the cell size has to be tuned with respect to the simulated model (i.e. the transmission range of the wireless nodes and the motion model). Such details of the simulated model are often unknown a priori or may depend on runtime characteristics (i.e. the variable transmission power).

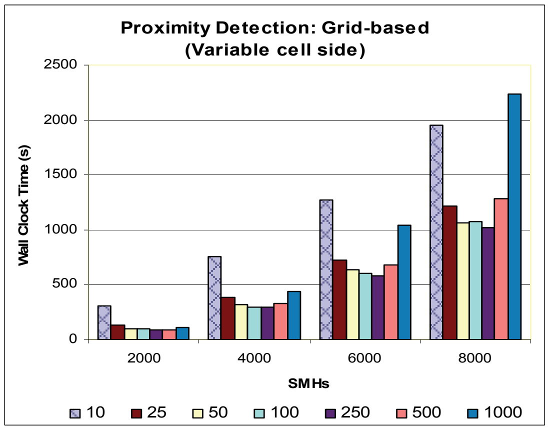

Figure 5. Performance evaluation of the grid-based approach, variable cell side (g=10..1000)

The mechanism based on results' caching for the reuse of distance computations, illustrated in section 4.3, has the results shown in figure 6. The WCT required with the *all-pairs* algorithm can be reduced, from 20% (2000 SMHs) to 33% (6000 SMHs).

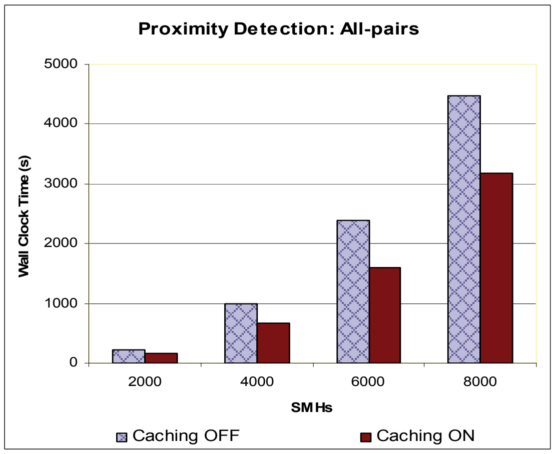

Figure 6. Performance evaluation of the All-pairs algorithm with Caching ON/OFF

Figure 7 shows the WCT reduction when composing various combinations of multistage filtering based on the previously defined proximity detection mechanisms. Basically, the results show that the incremental composition of multistage filtering schemes gives advantages, more evident when under high node density, as it would be expected.

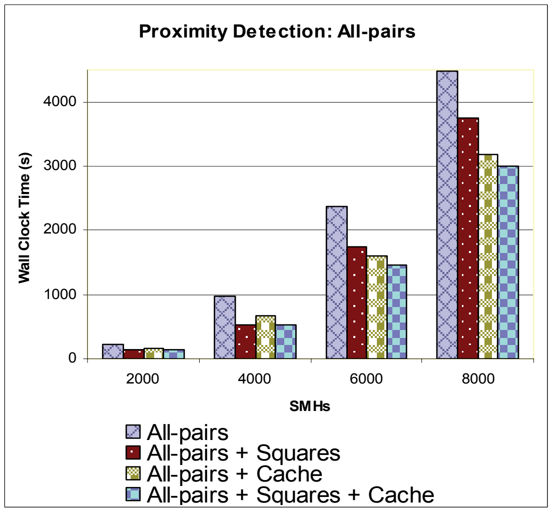

Figure 7. Performance evaluation of the All-pairs algorithm, combining different enhancements

The results presented in Figure 8, confirm that a migration-based approach can marginally increase the efficiency of the reuse-mechanism of cached results. By enabling the migration of SMHs, the highly interacting nodes are clustered together in the same LP, therefore increasing the probability to exploit pre-computed cache results. In this case the gain in terms of WCT appears as marginal. This is due to caching mechanism that has to be further optimized to reduce the runtime overhead, dissipating most of the gain obtained by the reuse factor.

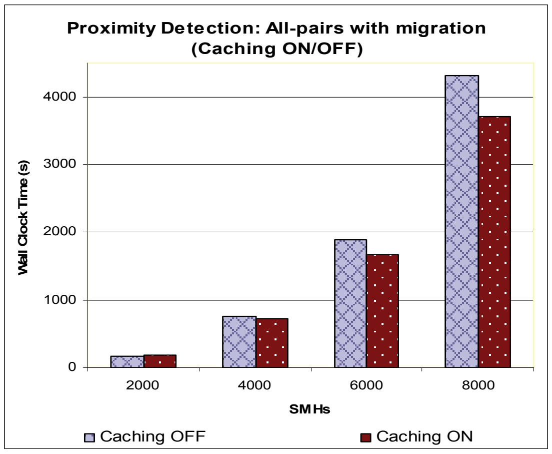

Figure 8. Performance evaluation of the All-pairs algorithm, Caching ON/OFF with migration enabled

As shown in [2,3] the migration based framework (GAIA) can reduce the simulation WCT, optimizing the SMHs allocation to reduce the amount of network communication. The performance enhancements introduced by GAIA have been widely investigated, but the interactions between a migration based approach and the proximity detection algorithms need to be carefully analyzed.

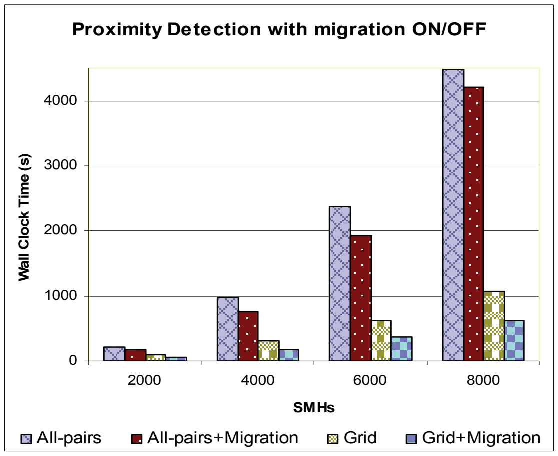

Figure 9. The impact of entities migration on the all-pairs and grid mechanisms

In Figure 9, the *all-pairs* and the *grid* algorithm are analyzed while turning ON and OFF the migration of simulated entities. As usual the simulation runs are repeated by increasing the number of SMHs. In this case both the *squares* and the results caching mechanisms are turned OFF. The results confirm that the migration of simulated entities reduces the WCT of both algorithms (*all-pairs* and *grid*-based). When applied to the *grid*-based, the migration mechanism can reduce the WCT up to 45%. In the case of the *all-pairs*, the simulation is computation-intensive, therefore the communication optimization, due to the migration mechanism, is less evident than the *grid*-based implementation.

Each migration is implemented as the transfer of some data, the allocation of a new entry in the data structures of the receiving LP and the de-allocation of the obsolete entry in the data structures of the sender. Therefore, migrations increase the overhead due to runtime insertion and deletion operations on the data structures managing the LPs internal state. On the other hand, Figure 9 demonstrates that the hash data structure used to implement the overlay grid is highly efficient and does not suffer relevant overheads when introducing the model entity migration scheme.

7. CONCLUSIONS AND FUTURE WORK

The proximity detection algorithms are a key part of the simulation of many mobile wireless systems. Many systems of interest, are complex enough that a monolithic execution architecture is unable to fulfill efficient and scalable simulation. A viable alternative based on the distributed simulation implemented on Grid Computing architectures is becoming attractive for the performance evaluation of complex systems. In this work we have analyzed some proposals for proximity detection algorithms being executed in a distributed architecture, with no shared memory support available. Some enhancements to the classical algorithms, and specifically tailored data structures have been proposed and evaluated. In addition, a model entity migration support has been composed with the simulation middleware to execute performance tests. Many interactions between the migration mechanism and some proximity detection algorithms have been investigated, resulting in guidelines about the opportune composition of migration mechanisms and enhanced proximity detection algorithms for distributed simulations of wireless mobile system models.

Future works will include improvements of proposed implementation, further investigation of proximity detection schemes, like adaptive grid cell-sizes. A related topic, collision detection, will be investigated as it could be seen as a special case of the proximity detection.

Acknowledgments

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

8. REFERENCES

[1] Blelloch, G., Narlikar, G. A practical comparison of N-body algorithms. *Parallel Algorithms. Series in Discrete Mathematics and Theoretical Computer Science,* Volume 30, 1997.

[2] Bononi, L., Bracuto, M., D'Angelo, G., Donatiello, L. Performance Analysis of a Parallel and Distributed Framework for Large Scale Wireless Systems' Simulation. *MsWIM 04: Proceedings of the 7-th ACM/IEEE*

International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems.

[3] Bononi, L., Bracuto, M., D'Angelo, G., Donatiello, L. A New Adaptive Middleware for Parallel and Distributed Simulation of Dynamically Interacting Systems. *DSRT '04: Proceedings of the 8-th IEEE International Symposium on Distributed Simulation and Real Time Applications*.

[4] Bononi, L., Bracuto, M., D'Angelo, G., Donatiello, L. Analysis of High Performance Communication and Computation Solutions for Parallel and Distributed Simulation. *HPCC '05: Springer LNCS Proceedings of the 2005 International Conference on High Performance Computing and Communications.*

[5] Bononi, L., Bracuto, M., D'Angelo, G., Donatiello, L. Scalable and Efficient Parallel and Distributed Simulation of Complex, Dynamic and Mobile Systems. *PERF '05: Proceedings of the IEEE FIRB-Perf Workshop on Techniques Methodologies and Tools for Performance Evaluation of Complex Systems.*

[6] Bononi, L., D'Angelo, G., Donatiello, L. HLA-based adaptive distributed simulation of wireless mobile systems. *PADS '03: Proceedings of the 17th ACM/IEEE/SCS Workshop on Parallel and Distributed Simulation*.

[7] Boukerche, A., Roy, A.J., Thomas, N. Dynamic Grid-Based Multicast Group Assignment in Data Distribution Management. *DS-RT '00: Proceedings of the Fourth IEEE International Workshop on Distributed Simulation and Real-Time Applications.*

[8] DMSO: Defense Modeling and Simulation Office. *High Level Architecture RTI Interface Specification*, Vers. 1.3, 1988

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

[10] Gao, J., Guibas, L.J., Nguyen, A. Distributed Proximity Maintenance in Ad Hoc Mobile Networks. *DCOSS '05: Proceedings of First IEEE International Conference on Distributed Computing in Sensor Systems.*

[11] IEEE STD 1516-2000. Standard for modeling and simulation, High Level Architecture (HLA).

[12] Ji, Z., Zhou, J., Takai, M., Martin, J., Bagrodia, R. Optimizing parallel execution of detailed wireless network simulation. *PADS '04: Proceedings of the eighteenth workshop on Parallel and distributed simulation*.

[13] Naoumov, V., Gross, T. Simulation of Large Scale Ad Hoc Networks. *MsWIM 04: Proceedings of the 5-th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems.*

[14] Perrone, L.F., Nicol, D.M. Using N-body Algorithm for Interference Computation of Wireless Cellular Simulations. *MASCOTS '00: Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*.

[15] Steinman, J.S., Wieland, F. Parallel proximity detection and the distribution list algorithm. *PADS '94: Proceedings of the eighth workshop on Parallel and Distributed Simulation*.

[16] Walsh, K., Sirer, E.G. Staged Simulation: A General Technique for Improving Simulation Scale and Performance. *ACM Transactions on Modeling and Computer Simulation (TOMACS), Special Issue on Scalable Network Modeling and Simulation*, 2004.

[17] Wieland, F., Carnes, D., Schultz, G. Using quad trees for parallelizing conflict detection in a sequential simulation. *PADS '01: Proceedings of the fifteenth workshop on Parallel and distributed simulation*.