Available online at www.sciencedirect.com
ScienceDirect
Computer Networks 52 (2008) 155–179
Computer Networks
www.elsevier.com/locate/comnet

MoVES: A framework for parallel and distributed simulation of wireless vehicular ad hoc networks

Luciano Bononi *, Marco Di Felice, Gabriele D'Angelo, Michele Bracuto, Lorenzo Donatiello

Department of Computer Science, University of Bologna, Mura Anteo Zamboni 7, 40127 Bologna, Italy

Available online 29 September 2007

Abstract

In this paper, we illustrate a Mobile Wireless Vehicular Environment Simulation (MoVES) framework for the parallel and distributed simulation of vehicular wireless ad hoc networks (VANETs). The proposed framework supports extensible, module-based and layered modeling, and scalable, accurate and efficient simulation of vehicular scenarios integrated with wireless communication and mobile services/applications. The vehicular layer includes models for vehicles, synthetic and trace-driven mobility, driver behavior, GPS-based street maps, intersection policies and traffic lights. The wireless communication layer currently includes models for physical propagation, and a network protocol stack including IEEE 802.11 Medium Access Control, up to the Application layer. MoVES provides a platform for microscopic modeling and simulation-based analysis of wireless vehicular scenarios and communication-based services and applications, like Intelligent Transportation Systems, communication-based monitoring/control and info-mobility services. The framework includes design solutions for scalable, accurate and efficient parallel and distributed simulation of complex, vehicular communication scenarios executed over cost-effective, commercial-off-the-shelf (COTS) simulation architectures. Dynamic model partition and adaptation-based load balancing solutions have been designed by exploiting common assumptions and model characteristics, in a user-transparent way. Test-bed performance evaluation for realistic scenarios has shown the effectiveness of MoVES in terms of simulation efficiency, scalability, adaptation and simulation accuracy.

Keywords: Wireless vehicular ad hoc networks; Parallel and distributed simulation; Intelligent transportation systems; Mobile services; Mobile communication

1. Introduction

Technology advances in portable devices, embedded systems and wireless networking have contributed to the marriage of mobile computing and communication for supporting new pervasive services and applications. As an example, implementation guidelines for a new class of mobile services for

* Corresponding author. Tel.: +39 051 2094972; fax: +39 051 2094510.
E-mail addresses: bononi@cs.unibo.it (L. Bononi), difelice@cs.unibo.it (M. Di Felice), gdangelo@cs.unibo.it (G. D'Angelo), bracuto@cs.unibo.it (M. Bracuto), donat@cs.unibo.it (L. Donatiello).

1389-1286/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2007.09.015

vehicle-drivers assistance and safety are the mission of worldwide Intelligent Transportation System (ITS) research programs [1,2]. The integration of positioning and multiple sensor capabilities, installed on-board vehicles [3], and specific wireless technologies for vehicular environment will enable car-to-car, car-to-infrastructure mobile communication and Vehicular Ad Hoc Networks (VANETs) [1,7,3,8–10]. These technologies include Dedicated Short Range Communication (DSRC) IEEE 802.11p Wireless Access for Vehicular Environment (WAVE) [4], and ISO TC204 Continuous Air interfaces-Long and Medium Range (CALM) [5,6]. VANETs can be considered as a mix of mobile ad hoc and sensor networks [3]: opportunistic communication up to intermittent/stable access to the Internet is possible, depending on the deployed wireless infrastructures. Distributed and coordinated communication-based services realized on top of VANETs are expected to improve safety (e.g., car accident reduction), real-time monitoring and equalization of traffic flows and parking resources, and the support of real-time info-mobility services about traffic jams, accidents, weather and dynamic travel route optimization [11,12]. Moreover, infrastructure-based inter-vehicle communication (IVC) systems may enable a wide set of vehicular killer applications, ranging from Internet access, e-payment and infotainment services.

A new generation of adaptation-based protocols and applications for VANETs are currently explored by some recent research projects headed by research institutes and car manufacturers [7–10]. The complex protocol architecture and convergence of wireless communications in vehicular systems is the subject of current research projects, e.g., the Cooperative Vehicle to Infrastructure System (CVIS) project [2] and the Network on Wheels (NOW) project [8]. In order to promote the introduction of novel applications based on IVC systems, it is essential to demonstrate the effectiveness of such protocols and service architectures over realistic environments by using accurate and fine-grained methodologies for the analysis [13]. Experimental characterization of IVC [14] may confirm the feasibility of ITS systems, under possibly different scenarios and service paradigms (e.g., client/server vs. P2P).

Computer simulation is used as a fine-grained and cost-effective support for analysis: the modeling of each area, vehicle and driver behavior can be accurate, and this would make it possible to evaluate both complex and hypothetical scenarios, which are impractical to realize as a real testbed. On the other hand, the adoption of simulation techniques for the analysis of large-scale, dynamic and complex vehicular networks, and related wireless communication-based services, creates severe problems in terms of simulation resources and simulation performance. Large data structures defining the state of complex model entities have an impact on memory requirements. Fine-grained simulation of urban traffic and wireless communication scenarios may easily involve thousands of causally interacting entities, each one characterized by local policies and attributes [15]. In a simulation, the individual evolution of a massive number of entities, the dynamic correlation effects due to mobility and to the local broadcast nature of the wireless transmissions, and the wireless signal propagation effects in urban scenarios [16] greatly increase computational requirements. Unfortunately, a sequential simulation of a real scenario with a large number of complex model entities (like vehicles running distributed services, and their wireless communications) may become computationally infeasible on a single physical execution unit (PEU) [17,18]. Solutions based on parallel computing architectures, like Single Instructions Multiple Data (SIMD) architectures have been considered in PARAMICS [19]: this approach may be useful under homogeneous modeling assumptions (that is, concurrently executed code updates different data structures each one representing the state of a homogeneous instance of the model entities). On the other hand, parallel and High-Performance Computing (HPC) architectures have high cost and require modelers to have expertise in parallel programming, to exploit the architecture potential. Parallel and distributed simulation (PADS) [20] offers a cost-effective technology to exploit concurrent computation over different PEUs. In this way, the simulation can be supported by low-cost aggregate computation and memory resources. In addition, the approval in 2000 of the IEEE 1516 Standard for Modeling and Simulation High Level Architecture (HLA) [6] has fostered the creation of middleware frameworks for PADS. Advantages of HLA-based PADS middleware include the support for integration of heterogeneous simulators, model reuse, and the solution of main simulation issues in a transparent way, from the user's viewpoint.

In this paper, we present a novel framework named Mobile Wireless Vehicular Environment

Simulator (MoVES), for the parallel and distributed simulation of urban mobility and wireless communication over vehicular ad hoc networks. A parallel and distributed simulation is implemented as a concurrent execution of Logical Processes (LP). In our approach, each physical processor runs an LP in charge of managing the execution of a subset of model entities: static entities like streets, traffic lights, intersections, and mobile vehicle entities with their wireless communication stack. In the simulation, interactions among model entities are implemented by exchanging time-stamped event messages (that is, message passing) between the corresponding LPs. If the message passing and synchronization overheads are designed and managed in a judicious way, a parallel or distributed simulation offers potential speedup thanks to the concurrent execution of computation tasks, and may result in high scalability at low-cost, thanks to the possible aggregation of computation, memory and communication resources of commercial off-the-shelf (COTS) systems. This would be useful for urban traffic simulation, as explored in [17,21,18]. The first version of the MoVES simulator presented in [22] was implemented from scratch as a concurrent execution of sequential simulators, coordinated by the kernel of a PADS middleware called Advanced RTI System (ARTIS) [23–25]. ARTIS is a middleware for generalized parallel and distributed simulation partially compliant with the High Level Architecture (HLA) standard IEEE 1516 [6]. In the first version of MoVES, each simulator was in charge of managing the coordinated execution of a portion of the simulation model. Initialization policies for uniform model allocation, load balancing, and entity migration functions were defined as a part of the MoVES framework. The main drawbacks of this approach were: the initial model partition, load balancing and entity allocation were based on static policies in the initialization phase, with some limitations that will be illustrated in Section 4, and the entity migration function was not completely user-transparent, being partially included in the model code. To cope with these limitations, the MoVES framework has been completely re-designed in this paper, with support provided by the ARTIS simulation middleware [23–25]. In the following, we will refer to the MOVES framework in general when discussing common features and implementation issues. More specifically, we will refer to the MOVES I implementation, and to the MOVES II implementation, to illustrate the differences between the first and second framework implementations, respectively. The new implementation of MoVES (MoVES II) defined in this paper has many additional advantages, illustrated in detail in Section 3, including user-transparent, dynamic and adaptation-based entity allocation and load balancing (both communication and computation), and synchronization and communication overhead reduction. This translates to simple model implementation, dynamic adaptation-based load balancing under generalized model assumptions, and simulation efficiency, scalability and accuracy, as shown in testbed performance evaluation for realistic model scenarios in Section 6.

The paper is organized as follows: Section 2 describes related work and existing solutions for PADS and monolithic simulation of vehicular and wireless systems; Section 3 illustrates the layered framework architecture designed for modeling and simulation of vehicular traffic and wireless communication; Section 4 illustrates the design and implementation of the proposed PADS architecture, including adaptation-based computation- and communication-load balancing and communication overhead reduction solutions; Section 5 sketches the definition of a testbed vehicular and wireless communication model; Section 6 illustrates some performance analysis results, based on realistic simulation scenarios (downtown area of the city of Bologna) to show the scalability and performance of the proposed PADS framework; Section 7 presents conclusions and future work.

2. Related work

Many vehicular-mobility simulation studies are based on fine-grained models that describe the behavior of individual entities instead of the behavior of (macroscopic) aggregate vehicular flows. Several simulators have been developed with various solutions to support both model complexity and scalability characteristics: VISSIM [26], PARAMICS [19,27] and SimTraffic [28] are some of the most successful commercial tools for simulation of complex, dynamic traffic scenarios, with high level of modeling detail, and support for realistic models, analysis and rendering. In general, to the best of our knowledge, such simulators do not include modeling support for wireless communication. In addition, the execution of simulation for very large vehicular models coupled with wireless radio

communication models, and network-based applications and services, may have a significant impact on scalability and performance issues. The integration of vehicular traffic and wireless communication models in VANET simulation environments have been proposed in recent works. CARISMA [10] is a traffic simulator which has been coupled with the Java-based AppSim tool to obtain a traffic and VANET simulation environment [29]. CORSIM [30] is a tool defined to model and simulate traffic in Atlanta, GA, which has been federated in a distributed simulation framework with Qualnet, a commercial wireless network simulator [31]. STRAW [13] is a traffic mobility and wireless network model simulated using the SWANS simulation tool [32]. A vehicular-mobility model for the network simulator (ns-2) is proposed in [33].

Only a few simulation packages currently support fine-grained event-based traffic models for PADS simulation. In some cases, the models are tightly-coupled with a PADS synchronization algorithms, by reducing the model portability and reuse. Alternative distributed implementations of PARAMICS have been proposed: in [17,34]; the divide-and-conquer concept is applied to the PARAMICS traffic simulation, by splitting the simulation map into sub-areas (called tiles), and executing the simulations of tiles on a cluster of PEUs connected by high-speed Ethernet links. Vehicle entities migration among different PEUs has been implemented to control synchronization overheads. A similar concept is presented in [18], in which synchronization among the LPs is managed by an external entity. HLA-based distributed simulation of traffic scenarios is considered in [21], and in [30]: a simulation is composed of federated components, each one executed in a common event-space with coordinated time-management scheme.

A set of simulation tools and models have been proposed to specifically assist in the modeling and simulation of wireless systems [35]. The list includes the open source monolithic network simulator (ns-2) [36], Omnet++ [37], and some powerful commercial tools like Opnet [38] and Qualnet [31]. In general, the scalability problem of massive wireless systems simulation has been investigated under PADS implementations in preliminary work [39,31]. The Parallel Distributed Network Simulator (PDNS) [40] is a parallel and enhanced version of the widely adopted Network Simulator (ns) [36], developed by the PADS research group at Georgia Tech. The simulator is based on the concept of federated simulation of multiple instances of ns schedulers. The Georgia Tech Network Simulator (GTNetS) [41] is a network simulation environment for large-scale networks. GTNetS supports the distribution of a single simulation topology on either a network of loosely coupled workstations, a shared-memory symmetric multiprocessing system, or a combination of both. The Dartmouth Scalable Simulation Framework (DaSSF) [42] is a C++ parallel and distributed simulation framework based on message passing (using MPI) running a combination of shared-memory and distributed-memory execution architectures. The new version (named iSSF, [43]) will support HLA interoperability, real-time simulation, and human-interaction capabilities. MAISIE [39] is a C-based simulation language proposed for sequential and parallel execution of discrete-event simulation of mobile wireless systems. The Telecommunications Description Language (TED) [44] is an object-oriented simulation language that can be used to model communication systems, supported by the Georgia Tech Time Warp (GTW) [44], an optimized parallel simulation kernel for parallel and distributed simulation. TED has served as the basis for the implementation of WiPPET [45], a parallel simulator designed to model the radio propagation, mobility and protocols of wireless networks. In SwiMNet [46], mobility is assumed to be not correlated to wireless communication: a pre-computation of the mobility patterns of mobile hosts is done to know in advance the effects of mobility, and to optimize parallel and distributed simulation of wireless PCS networks. GloMosim [47] is a simulation environment for wireless and wired network systems based on the Parsec parallel discrete-event simulation kernel. QualNet [31] is a complete and efficient commercial tool derived from GloMoSim. It supports both monolithic simulations and parallel simulations. It is reported to scale up to 10's of thousands of nodes and it includes a complete and detailed library of protocol models for wireless systems. Qualnet options include the support for HLA & Threaded Communication Modules. SWANS [32] is a Java-based scalable wireless network simulator built on the top of JiST, a high-performance discrete-event simulation engine. OPNET [38] is a commercial framework for modeling and simulation of wired and wireless communication systems. It includes a large library of implemented protocol models. The tool supports discrete-event, hybrid and analytical simulation; it runs in both sequential and parallel

simulation mode, and supports HLA and co-simulation technologies.

A summary of motivations, new characteristics and advanced methodologies specifically implemented in the MOVES framework for the parallel and distributed simulation of VANET scenarios is reported in Section 3.1.

3. The parallel and distributed simulation architecture

In this section we illustrate the aims, the motivations, and the logical structure of a PADS architecture for modeling and simulation of wireless vehicular networks. Our PADS framework has the following aims:

- Scalability: by exploiting aggregate computation and memory resources of different PEUs, simulation of massive models can be supported;
- Transparency: by exploiting PADS primitives implemented by the ARTIS middleware, the distributed model implementation should be as transparent as possible to the modeler;
- Adaptation-based overhead reduction and load balancing: by exploiting PADS primitives and load balancing heuristics implemented by the ARTIS middleware, we defined adaptation-based solutions to initialize, allocate and migrate model entities over COTS simulation architectures. Dynamic adaptation improves the utilization of shared resources and the simulation performance by reducing message passing and synchronization overheads between LPs;
- Software modularity and reuse: the simulation tool and modeling architecture allow the composition and reuse of module-based model components, including modeling and simulation management modules.

3.1. Motivation and previous work

A number of available vehicular traffic simulators (i) are commercial products, (ii) do not support distributed/parallel simulation, or (iii) do not include models for the wireless networking simulation. A federation of existing traffic and wireless simulators, like the ones described in Section 2, is possible only if all simulators implement a common interoperability standard, like IEEE 1516 [6]. In general, even if the performance and compatibility characteristics of federated simulators are good, the performance of such an approach may be poor due to: (i) sub-optimal utilization of resources, (ii) lack of a coordination scheme for resources adaptation and load balancing, and (iii) lack of exploitation of model characteristics and assumptions. For these reasons, we developed a novel framework (named Mobile Wireless Vehicular Environment simulator, MoVES) for the fine-grained (microscopic) modeling and simulation of realistic traffic scenarios, wireless networks and communication-based services and applications. The first implementation of the MoVES framework (MoVES I in the following) proposed in [22] was realized as a self-contained simulation framework, based on parallel execution of coordinated sequential simulators. Each simulator was synchronized by exploiting the ARTIS parallel and distributed simulation middleware [24]. Many assumptions, and runtime optimization solutions, were implemented at the modeling framework layer. The MoVES I implementation has some drawbacks: (i) the optimization approach is based on strong assumptions, (ii) the optimization functions are model-dependent, and (iii) the optimization functions are local to each sequential simulator, and may lead to sub-optimal global performance. Details about these issues will be illustrated in Sections 4.3.2, 4.4.1 and 6.1. In this paper, MoVES has been completely re-designed and re-implemented on a layered PADS architecture (see Fig. 1). A new implementation of the MoVES framework (MoVES II in the following) is now realized as a distributed simulation, based on additional ARTIS middleware functionalities. This new layered approach has potential advantages, illustrated in the following Section.

3.2. The PADS architecture description

By looking at the PADS architecture (Fig. 1), three separate layers are defined: the MoVES framework layer, the ARTIS middleware layer, and the physical execution units (PEUs) layer.

Starting from the bottom of Fig. 1, the PEU layer represents the physical computation and communication system supporting the parallel and distributed simulation. In general, a physical execution unit (PEU) is defined as an autonomous processing unit, connected to other PEUs by network hardware. A PEU may have shared-memory multiprocessor architecture and multi-threading CPUs, or a single processor architecture. In our approach, each physical processor runs a Logical

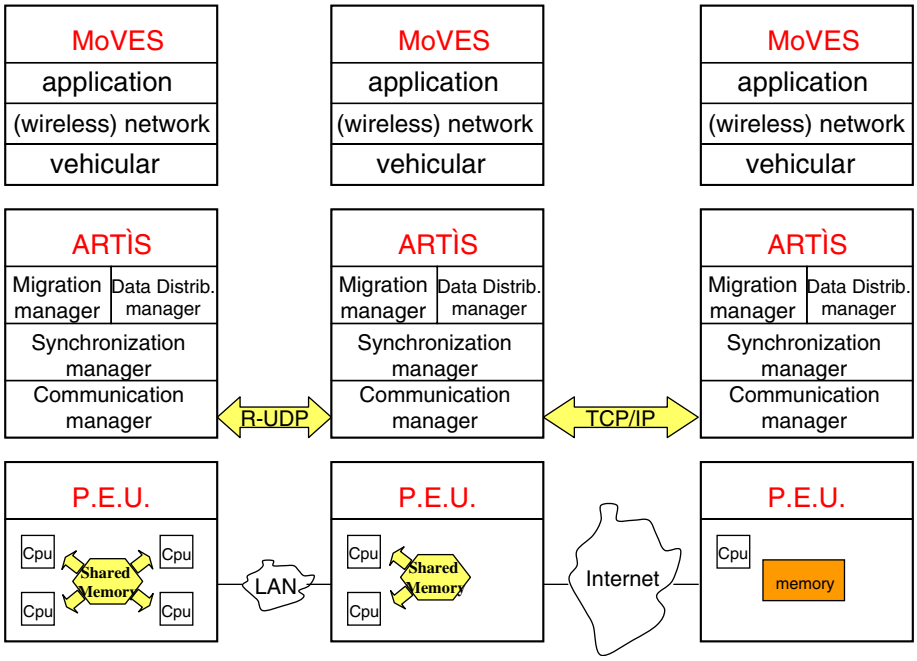

Fig. 1. The PADS simulation framework architecture.

Process (LP) in charge of managing the execution of a subset of model entities. In general, a single Logical Process execution cannot be split over multiple CPUs. Each CPU can allocate many different LPs. The communication support is provided by shared-memory message passing between LPs on the same PEU, and by LAN network communication up to Internet-based communication, between LPs on different PEUs.

The ARTIS layer includes the middleware for the management of parallel and distributed simulations over heterogeneous execution architectures. ARTIS is written in C, it provides bindings for C/C++ and Java modeling frameworks. It implements many solutions to support and to optimize the parallel and distributed simulation of massive, dynamic and complex system models over heterogeneous execution architectures [25]. ARTIS design and implementation is still an ongoing activity, and more details can be found in [24,25]. For the purposes of this paper, the ARTIS middleware is sketched for illustrating the support provided to the MoVES framework. The ARTIS middleware provides service APIs to MoVES, associated with the functions implemented in middleware modules. The ARTIS Communication Manager implements scalable communication functions between LPs, see Fig. 1. This is obtained by exploiting data marshalling solutions and selection of efficient network protocols for the physical communication layer. As an example, the communication is implemented by low-overhead message passing between CPUs with shared memory, Reliable-UDP between PEUs connected by reliable high-performance LANs, and TCP/IP between remote PEUs connected by the Internet. The Synchronization Manager implements policies to coordinate the events execution and state updates of distributed LPs, that is, to ensure that simulated events are processed in causal order. The Data Distribution Manager (DDM) module is in charge of managing the data distribution among LPs implementing a parallel and distributed simulation. In our approach, the distributed simulation architecture is based on a global sharing of a subset of model state information (as an example, the position of vehicle model entities). Appropriate DDM techniques are implemented to reduce the update cost of the shared information. This is a major difference with respect to the MOVES I implementation in [22], realized as a coordinated execution of sequential simulators, without information-sharing support. The new approach introduces controlled communication overheads, due to the additional amount of communication required when periodic state sharing is performed. On the other hand, the new approach introduces advantages in (i) the sim-

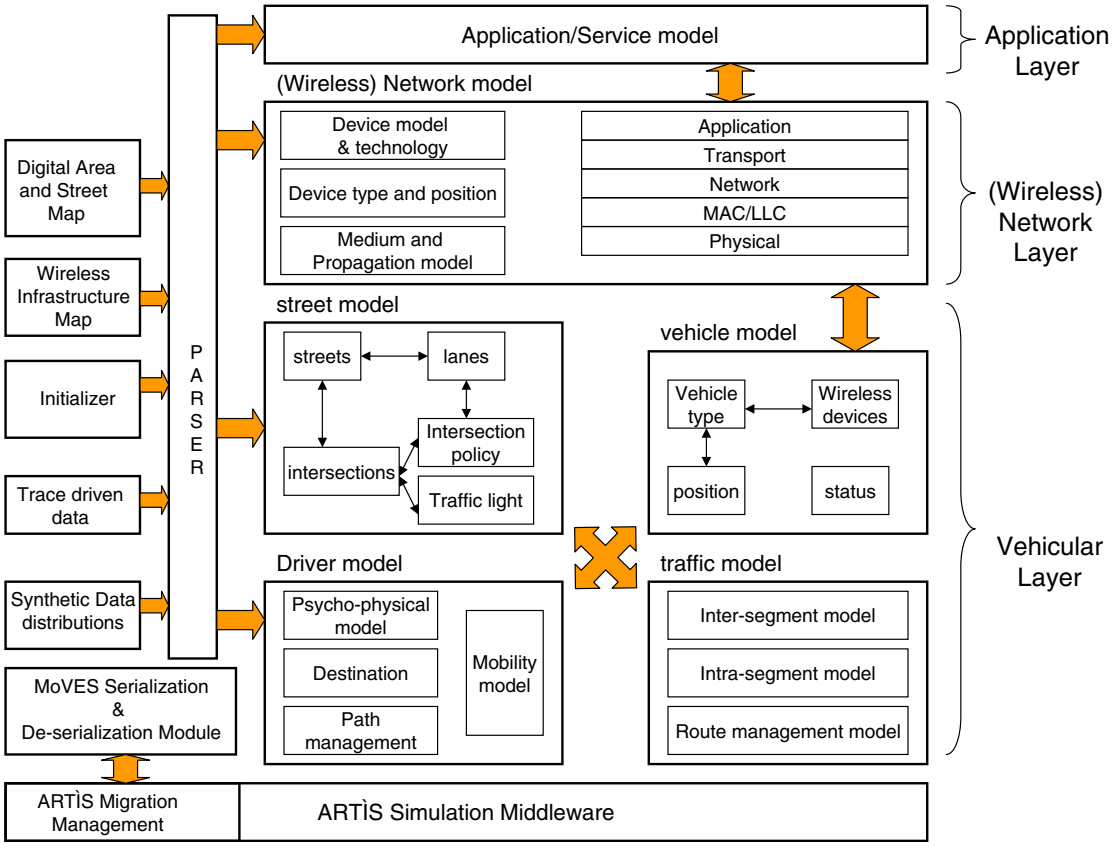

Fig. 2. The MoVES layer architecture.

plification of real-time monitoring functions, (ii) the exploitation of ARTIS load balancing heuristics, in a way transparent to the MoVES framework, (iii) performance gain, under realistic scenarios and assumptions, as shown in the performance evaluation Section 6.1. The Migration Manager module implements different functions, including communication- and computation-load balancing based on runtime migration of model entities between the PEUs. The ARTIS migration implementation and load balancing heuristics are general and parameter-based. They have been carefully designed to reduce overheads and to ensure adaptation to the model and to the simulation architecture characteristics, as will be described in Section 4.

The MoVES layer is written in Java, and it includes three modeling sub-layers: (1) the application sub-layer allows the user to model communication-based services and applications, (2) the networking sub-layer allows the user to model wireless (and wired) technologies and protocol architectures, network infrastructures (if any), static devices, medium and signal propagation characteristics, and (3) the vehicular sub-layer allows the user to model vehicles with and without mobile network devices, realistic street topologies, driver behavior, traffic scenarios and traffic management policies. A detailed architecture of the MoVES layer is shown in Fig. 2. The MoVES application sub-layer includes models of communication-based services and applications. Services and applications running on top of virtual devices can be easily modeled by using an approach based on socket-like APIs provided by the model of the network protocol suite. At the Network sub-layer, MoVES implements the model components defining a protocol stack for network communication devices. Device technologies, and static device infrastructures can be defined, if any are needed. The Vehicular sub-layer provides extensible modeling of realistic traffic scenarios, and mobile wireless devices installed on vehicles. Specifically, street model components include streets, lanes, intersections, intersection policies and traffic lights. Vehicle model components may

include wireless devices. Driver model components include psycho-physical attitude, mobility model and path preferences. The traffic model components include route management policies, intra- and inter-segment models, sketched in Section 5.1. In addition, the MoVES layer includes service modules: a parser component allows initializing and feeding model entities with data obtained from digital GPS maps, wireless infrastructure maps, and synthetic and trace-driven data sources. The serialization and de-serialization module is in charge of serializing (as byte-streams) the state information of model entities, to be migrated between different PEUs by ARTIS. Not shown in Fig. 2, a logging and data monitoring utility supports saving and analyzing traces and pre-defined performance figures. A prototype rendering application supports trace animation (as an example, car mobility and wireless transmissions), both at runtime, and offline.

4. Design and implementation of MoVES framework

In the following we illustrate the design and implementation of the PADS architecture for modeling and simulation of wireless vehicular networks described in Section 3. Verification and validation of the proposed simulation framework and related models have been performed during the design and implementation phases, and have demonstrated the correct functionality of the simulation execution platform, and the expected degree of model accuracy.

The simulation framework proposed in this paper may potentially support different tasks under a layered modeling architecture (see Fig. 2). At the vehicular layer, (i) modeling and analysis of realistic, detailed areas of interest, including GPS-based street maps, lanes, intersections, traffic lights and related policies, (ii) modeling of vehicles and driver psycho-physical attitude, (iii) modeling of synthetic or realistic trace-driven mobility models [15], traffic patterns and positive/negative attraction points (like dense work areas at 8.00am and 6.00pm, respectively), (iv) modeling of traffic policies [19,13,29,26] and the impact of "what if" management assumptions [48]. At the wireless communication layer: (v) modeling of wireless communication scenarios, supported by opportunistic VANETs and infrastructure-based communication networks [13,14,48], wireless propagation models, and a network protocol stack including physical and MAC technology libraries (e.g., IEEE 802.11) up to the application layer. At the application layer, simulated processes of target applications/services can be virtually immersed in the simulated scenario.

4.1. Time-stepped synchronization scheme

Different PADS communication and synchronization solutions have been proposed. In this work we consider a conservative time-stepped synchronization scheme [20]: simulated time t is advanced by a time-step only after events scheduled at time t have been managed. A synchronization phase is needed to deliver new event-messages and to update model entities in order to maintain a causal order of events before the next time-step execution. This synchronization, managed by the ARTIS Synchronization module, is one overhead factor of the PADS approach. We selected the conservative time-stepped scheme because it is simple, it offers the opportunity to implement scalable solutions for multi-layered models, and to implement migration-based load balancing schemes specifically designed to exploit the characteristics of the considered models. Optimistic solutions [20] would need costly state-saving management, which would not be appropriate under the migration scheme we adopted for load balancing in this middleware framework. In addition, the time-step duration can be adapted by the simulation middleware, depending on the granularity of events of interest for the analysis. As an example, in a vehicular traffic simulation, the time-step could be quite large, since the granularity of events can be assumed in the order of a tenth of a second (e.g., the reaction time of a vehicle driver). In a simulation of a wireless scenario the time-step could be in the order of microseconds (e.g., a slot-time of IEEE 802.11 Medium Access Control protocol). When both models are coupled under the same simulation framework, as an example, to study the effects of mobility on wireless communication, or the effect of wireless communication info-services on the mobility of vehicles, the time-step length could be dynamically managed by the simulation middleware. This enhancement is considered a future work activity for the MoVES framework.

4.2. Simulation performance issues

Beyond scalability, one motivation of the PADS approach is to obtain simulation performance speedup. The main performance problem of a parallel and distributed simulation is to obtain the maximum advantage of concurrent computation of

events, while reducing communication and synchronization overheads needed for PADS coordination. The speedup can be measured as the ratio between the amount of Wall Clock Time (WCT) required for completing the same simulation test under different simulation approaches. Speedup is a normalized index of the overall time required for simulation: speedup values greater than one imply a gain in simulation performance. A speedup can be obtained by increasing hardware performance, by increasing the degree of computation parallelism (that is, the number of PEUs), and by reducing the computation, communication and synchronization overheads. Unfortunately, from a speedup viewpoint, tradeoffs must be considered in the implementation of PADS: in general, the adoption of high-performance hardware is costly, and the increase on parallelism translates in high communication and synchronization overheads. The optimal balancing of speedup tradeoffs is hard to define a priori, because dynamic factors bias the optimal balancing at runtime. As an example, the rate of interactions between model entities, and the performance of hardware communication resources (e.g., latency of a shared LAN) may show unpredictable variations during the simulation execution, biasing the balancing of communication and synchronization overheads. The amount of computation required to manage simulated events and model state updates, and the utilization of computation resources (e.g., CPU load of shared multi-programmed PEU architectures) can vary in an unpredictable way during the simulation execution, biasing the load balancing of computation resources. For these reasons, in the proposed middleware, we do not assume the system and network resources to be allocated for the simulation in an exclusive way. In addition, we considered two classes of generalized and low-cost solutions to improve the simulation efficiency: (i) communication- and computation-overhead reduction, and (ii) dynamic communication- and computation-load balancing.

4.3. Communication and computation overhead reduction

We summarize two approaches that have been followed for overhead reduction in preliminary and in current implementation of MoVES.

4.3.1. Terminology

For the sake of clarity, we define some formal terminology. We call a "*simulated area*" or simply "*area*" the model of the space populated by simulated entities. As an example, in our experiments, the simulated area is the downtown area of the city of Bologna. A simulated area can be partitioned in regular units called "*cells*". An arbitrary collection of adjacent cells is called a "*region*". A "*mobile entity*" is a model entity which moves in the simulated area (as an example, a vehicle). The concept of "*model entity migration*" is the transfer of (static or mobile) model-entity's data-structures and management from one "*home-LP*" to another "*foreign-LP*". Based on the communication architecture connecting the PEUs, we define three classes of connectivity for LPs: "*SHM-connected LPs*" are LPs interacting via (very fast) shared memory, "*LAN-connected LPs*" are LPs interacting via (fast) LAN technology, and "*I-connected LPs*" are remote LPs interacting via (slow) Internet connections. The concept of "*space proximity*" of model entities indicates that the model entities are near to each other (that is, Space-neighbors, "*S-neighbors*") in the simulated area. The concept of "*allocation proximity*" of model entities (that is, Allocation-neighbors, "*A-neighbors*" model entities) is intended as the fact that model entities are allocated (and efficiently communicating) on the same "home-LP", or on tightly-coupled SHM-connected LPs. A "*Causal Set*" (*C-set*) for event E is the collection of model entities causally influenced by the occurrence of event E. As an example, the *C-set* of a wireless transmission event contains all the wireless receivers in the transmission range.

4.3.2. MoVES I: Communication and computation overhead reduction

The MoVES I implementation was based on a coordinated execution of concurrent simulators. Each simulator was implemented as a single LP executed over a single CPU. Each LP was assumed to manage the set of model entities positioned in a statically defined region of the simulated area. Computation overhead reduction was based on the use of efficient data structures and smart algorithms for event-management and the state-update of model entities. Communication overhead reduction solutions were implemented to reduce the cost (latency) of communication and synchronization between LPs. To this end, we assume that the hardware architecture supporting LP communication is composed of communication resources with different performance: as an example, shared memory between LPs on the same PEU, local area network

technology between a local cluster of PEUs, up to Internet communication between remote PEUs (see Fig. 1). In general, the support of a PADS middleware includes message passing primitives, for notifying events and updates to LPs in charge of managing the simulation of model entities. As an example, an event-message notification is needed between two different LPs A and B, when the simulated event generated by a model entity X, managed by LP A, has a causal effect on at least one model entity Y managed by LP B.

In wireless and vehicular models, the amount of event correlation and causality between simulated model entities depends on their "space proximity". In other words, the *S-neighbor* entities in the simulation area show a strong causal correlation of events. We illustrate this concept with an example taken from our model of interest: in a scenario where a wireless transmission event is originated by a vehicle, the event influences all the *S-neighbor* vehicles positioned in the simulated area, within the wireless transmission range. In other words, the *C-Set* of wireless transmission events includes a significant subset of the *S-neighbor* entities of the transmitting vehicle. A possible solution to reduce communication overhead is to send event-message notifications only to destinations in the *C-Set* of the generated event. In general, we can assume this result is accomplished by well-designed PADS middleware solutions for Data Distribution management. One additional solution to further reduce communication overheads is the following: the *C-Set* of events should be managed by a minimal number of LPs (possibly, one LP), and such LPs should belong to the class of the best connected communication architectures, like SHM-connected LPs. If this condition could be maintained, few efficient communications between a few LPs would be sufficient to satisfy all the event-communication requirements.

Intuitively, one solution is to allocate the model entities on the LPs, and the LPs on the PEU architecture, such that the maximum number of *S-neighbor* model entities in the simulated area become *A-Neighbor* model entities, that is, entities managed over the best connected LPs. This would reduce the communication overhead. To implement this solution, in the MoVES I implementation, we defined three static policies for dividing the simulated area in cells, and to cluster adjacent cells into regions (see Fig. 3).

The regions assigned to the LPs are defined by aggregating adjacent cells with three possible policies: (P1) Equal Stripes approach, which aggregates the map cells into a set of stripes with the same area (number of cells), (P2) Balanced Stripes approach, which produces different regions (stripes) containing the same number of intersections, and (P3) Greedy Approach, which creates macro-cells of clustered adjacent cells by equalizing the number of intersections in efficient way. The greedy algorithm produces an equalized N-partition of the map (being N the number of LPs) with a cell-merging approach described in [22].

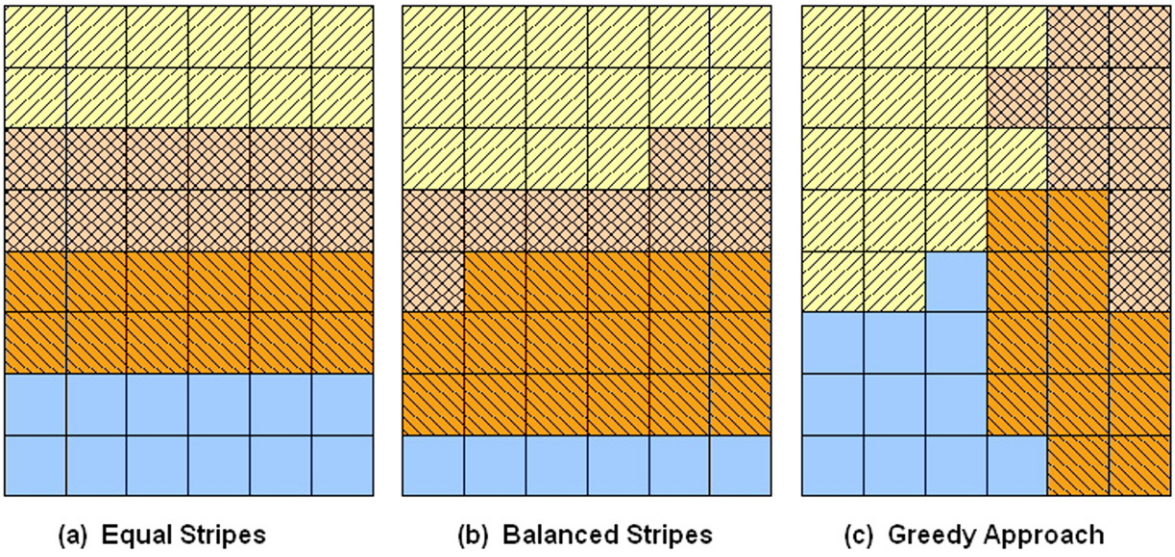

Fig. 3. Example of map partitions created by three different area mapping algorithms used in MoVES I.

In this approach, each intersection and vehicle entity was assumed as a homogeneous computation unit. By balancing the number of crossings in the regions, and by assuming a uniform distribution of vehicle entities in the area, both balanced stripes and greedy algorithms would create map-dependent, computation-balanced partitions. Such final partitions (called regions) are depicted as colored areas in Fig. 3.

To summarize, each region is statically allocated on one single LP, such that a high number of *S-neighbor* model entities are allocated as *A-neighbors* model entities over SMH-connected LPs. Unfortunately, this scheme has to cope with two problems: (*p1*) since each LP only manages entities in its allocated region, mobile entities crossing the region boundaries have to be migrated to a different LP managing the destination region, and (*p2*) any static ideal allocation of regions (and their model entities) on LPs, based on "space proximity" communication-overheads optimization, would be quickly disrupted due to mobile entities changing their position. As an example, a mobile vehicle X initially positioned in a cell of an urban area, managed by LP A at simulated time t, may change position at time $(t + k)$ by approaching the area boundary. In this way, the mobile vehicle entity could be "surrounded" by foreign model entities managed by the foreign LP B. If the LP B is allocated on a PEU with low communication performance, this would originate a flow of external event-messages dynamically increasing the communication overheads.

The migration of model entities was defined as a service of the MoVES I implementation [22], primarily because it was needed for implementing the coordinated simulators execution (see the problem *p1* above). Migration was implicitly used as a communication overhead reduction solution, because the mobile entity migration maintains the clustering of the S-neighbor entities under the same LP (that is, contributing to solve the problem *p2*).

The migration policy of MoVES I was not based on heuristics. It was model-based, with a simple and straightforward definition: when a mobile entity (that is, a vehicle) crosses the area boundary managed by its own home-LP, the model entity migration is performed to the destination-LP.

4.3.3. MoVES II: Communication and computation overhead reduction

Some facts can be summarized before proceeding with the illustration of the MoVES II implementation guidelines: (i) in the MoVES I implementation, the migration scheme was model- and area- (region) dependent, and only mobile entities crossing the area boundaries were migrated; (ii) model causality in wireless vehicular models may be considered as an abstraction of the "spatial proximity" concept; (iii) mobile entities, like vehicles in simulated vehicular scenarios, dynamically change their S-neighbor set during the simulation, and in this way, (iv) communication overhead reduction schemes based on a static association between LP and pre-defined regions (approximating a clustering of the S-neighbor sets), will see a decrease in their effectiveness, eventually; (v) dense event-generation is produced by wireless communication scenarios, compared with traffic simulation scenarios, hence the wireless scenario assumptions may have a dominating effect on cost reduction policies. All these guidelines have been considered in the MoVES II implementation, as illustrated in the following.

In the MoVES II implementation, the model entity migration has been made almost transparent to the modeling layer: only the serialization and de-serialization of migrating entities (data structures) are provided by MoVES, and migration primitives are implemented by the ARTIS middleware. In addition, in the MoVES II implementation, the migration scheme has been extended to all model entities (that is, not only the mobile ones). Moreover, the assumption of a static allocation of area regions on the LPs has been relaxed. The new approach allows extending the use of the entity migration functionality, as a dynamic adaptation scheme, under the control of multiple heuristics defined in the ARTIS middleware. This can be exploited to perform dynamic communication overhead reduction, and dynamic computation- and communication-load balancing. The new MoVES II implementation allows relaxing many assumptions made in the first implementation, including the fact that now migration can be triggered by heuristics, and not only triggered by entities crossing static area boundaries. This allows defining new adaptation-based policies, for clustering model entities on LPs under more general and adaptive communication reduction guidelines, as an example, based on "spatial proximity" and "interaction rate". Fig. 4 illustrates an example of the new approach implemented in MoVES II to realize migration-based adaptive communication-overhead reduction. By looking at Fig. 4, the MoVES simulated area shows the position of mobile vehicles

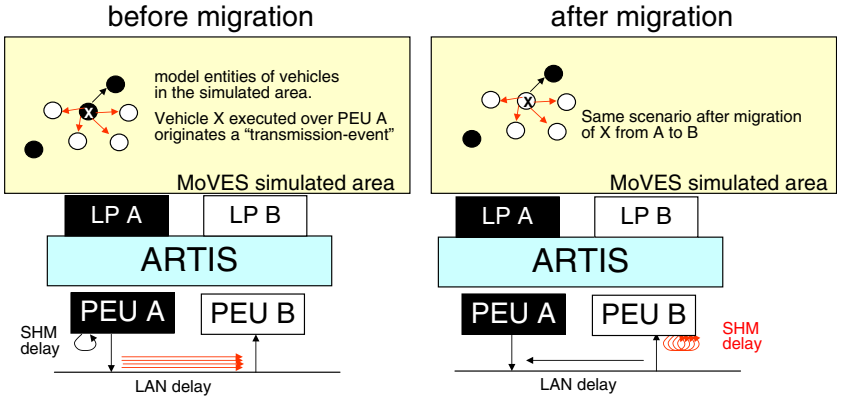

Fig. 4. The MoVES II migration based on ARTIS heuristics for communication overhead reduction.

(as dots). The color of dots identifies the LP (and the corresponding PEU) managing the vehicle entity. Black vehicles are managed by LP A on PEU A, and white vehicles are managed by LP B on PEU B. In this example, we assume that the allocation of black and white vehicles was initially defined in an optimal way, based on a static area-partition and a static allocation-scheme (that is, all white/black vehicles were in the proximity of each other and all white/black vehicles were managed on LP B/A). Anyway, the simulated mobility of vehicles could quickly produce a scenario like the one depicted in Fig. 4, during the simulation. In Fig. 4(left) the X vehicle is surrounded by white vehicles. The X vehicle simulates a wireless transmission event which must be notified as an event-message to all the S-neighbors entities within the transmission range (that is, the *C-set* of the transmission event generated from X, denoted with thin arrows departing from X). One local event notification message is exchanged via shared memory between X and the other black vehicle, implemented on the same LP, in Fig. 4(left). The remaining 4 external event notification messages will reach the external LP B on PEU B, being transmitted via LAN technology. The Local Communication Ratio (LCR) is defined as the ratio between the number of communications internally performed by an LP, with respect to the total of communications of the same LP. Intuitively, the LCR illustrates how the migration-based clustering of model entities is able to reduce the communication overhead of the simulation. The LCR for the event management in Fig. 4 is defined as $LCR = 1/(4+1)$ on the LP A managing X. This means that low-overhead communication is exploited only with 20% probability by LP B. If we assume that a migration-based heuristic is defined to implement a communication overhead reduction, the migration of vehicle X could be implemented from LP A on PEU A, to LP B on PEU B. This would lead to the scenario depicted in Fig. 4(right). In this case, the new LCR is obtained as $LCR = 4/(1+4) = 80\%$ for the same scenario, and for future execution of events originated by X on LP B. The idea of maintaining highly-interacting neighbor entities under the management of the same LP can be implemented, by migrating model entities between LPs at run-time, based on dynamic estimates like the LCR. In this way, the mobile vehicle X could be dynamically migrated on the new home-LP B, which includes a majority of interacting entities (see Fig. 4, right).

Figs. 5 and 6 show a visual snapshot of the traffic simulation in the city of Bologna, by illustrating the effect of the MoVES II migration scheme. The street map shows vehicles positions as dots, and colors indicate which LP manages the vehicle entity simulation. Fig. 5 shows the initial random allocation of vehicle entities over three simulation LPs. Fig. 6 shows the effects of the MoVES II migration based on ARTIS heuristics for communication overhead reduction obtained after a few simulated time-steps. The migration scheme keeps vehicles clustered, by maintaining highly-interacting neighbors under the management of the same LP.

The new migration scheme is also adopted for load balancing mechanisms in MoVES II, under general heuristics, assumptions and scenarios, as illustrated in the following Section 4.4.2.

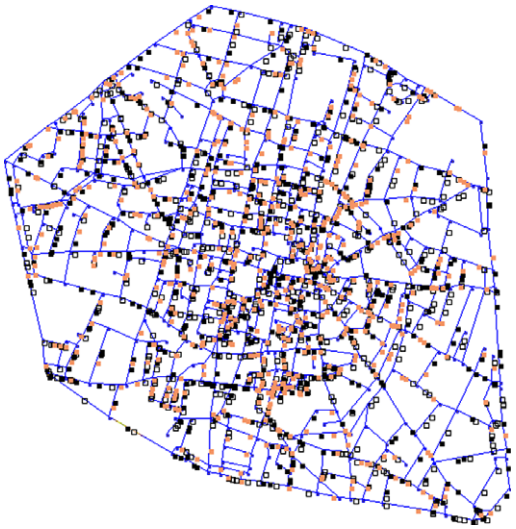

Fig. 5. Initial random vehicles allocation over three simulation LPs (dots colors indicate LPs).

Fig. 6. Runtime allocation of vehicles over three simulation LPs, under the effect of MoVES II migration scheme.

4.4. Dynamic communication- and computation-load balancing

In this section we shortly summarize two approaches that have been followed for dynamic communication and computation load balancing, in the MoVES I and in the MoVES II implementations, respectively.

4.4.1. MoVES I: Dynamic communication and computation load balancing

In the MoVES I implementation [22], some model and system assumptions were considered as the basis for the proposed computation-load balancing solution. In MoVES I, each LP was assumed to manage a statically defined region of the whole simulated area. The main assumption concerning load balancing was that the LP's computation load was proportional to the size of respectively simulated regions. More specifically, we assumed that the number of computation-intensive entities in the region managed by an LP, like street-crossings and vehicles, were statically balanced in the initialization phase of model allocation. Three initial area-allocation policies were investigated (see Section 4.3.2) to find solutions for static computation-load balancing in [22]. In addition, computation-load balancing was based on the assumption that mobile model-entities (vehicles) had homogeneous computation cost and uniform distribution in the simulated area. Pseudo-random mobility models, like the Random Waypoint, were used to maintain the assumptions on uniform distribution of mobile model entities. As a consequence of the above assumptions, the initial region-allocation phase (Section 4.3.2) was implicitly a static policy for computation-load balancing between the LPs. It is worth noting that CPUs and memory resources supporting the execution of simulation LPs were assumed to be homogeneous. These assumptions may be difficult to meet in real simulation system implementations and scenarios. As an example, at the system level, PEUs with heterogeneous performance and variable background load would violate the assumptions considered. The same violation may occur if the hot-spot traffic scenarios (like congested traffic areas) and attraction points are present in the model.

4.4.2. MoVES II: Dynamic communication- and computation-load balancing

MoVES II required a significant re-implementation effort: a distributed simulation framework, managed by the ARTIS PADS middleware, has replaced the coordinated execution of independent simulators realized in MOVES I. The main difference is that the middleware functions now implement a shared state of the whole simulation on every LP. This fact has advantages and drawbacks. Advantages include the possibility to implement simulation monitoring functions on each LP, and new middleware solutions for overhead reduction

and load balancing, based on general assumptions. The main drawback is the additional amount of communication overhead needed to maintain the shared state information. As we will show in the design illustration and performance evaluation sections, under worst case scenarios, advantages will balance the overhead of this new implementation.

In the new implementation, a dynamic computation-load balancing is controlled by the heuristics defined in the ARTIS Migration module. The implemented approach is simple and general, in order to capture the essence of the problem at the middleware layer, without critical assumptions. Each LP is monitored during the simulation by the load balancing heuristics. The heuristics implementations are discussed in Section 4.5. In a time-stepped synchronization scheme, like the one adopted in this framework, unified computation- and communication-load balancing can be defined as follows: each LP must complete a round of computation and communication (synchronization) with the other LPs, at the same Wall Clock Time. Intuitively, it is important that all LPs get synchronized for advancing to the next time-step of simulated time, with the minimum Wall Clock Time difference. In fact, all LPs advance the simulated time synchronously with the slowest LP. If one LP is early with respect to others reaching the synchronization barrier, this means that some more computation load could be used to fill the gap and to alleviate overloaded LPs. On the other hand, if one LP is late with respect to others, its computation overload should be eliminated. It is worth noting that the above condition is sufficient for load balancing, without discriminating between computation- and communication-resources performance. In other words, the real achievement is not given by a symmetry in the utilization of resources, and by the utilization of all available resources, but is given by the homogeneous result in the composition of two costs: the computation and the communication needed for performing the parallel and distributed simulation. This approach can be effective on a COTS, heterogeneous, shared and distributed computation and communication architecture like the one we assume for our testbed simulations. One of the best results achievable with this scheme is that the distributed simulation architecture can proactively allocate new computations on more available PEUs, or dismiss adopted PEUs. The proactive management policy depends on the fact that allocated PEUs positively or negatively contribute in the tradeoff between concurrent execution and communication overheads. As an example, computationally efficient PEUs could be dismissed because their communication latency is slowing down the overall system performance. Conversely, one CPU with only 10% residual CPU time could be included because its small computation potential is effective when coupled with highly efficient communication architectures.

4.5. ARTIS migration heuristics

A detailed illustration of the ARTIS heuristics can be found in [23,25]. In this work we sketch the implementation of heuristics used for controlling migrations. Migrations are the basis for load balancing and communication overhead reduction solutions implemented in MoVES II. MoVES II allows the migration of all model entities. In other words, in MoVES II, the LPs are no longer statically initialized and associated with regions. Now, the LPs can incrementally migrate a set of entities, under the control of migration heuristics. This would lead to an adaptive clustering of model entities, whose allocation on the LPs is independent with respect to entity positions in the simulated area, and completely based on the ''entity interaction'' concept, as shown in Fig. 4. The set of model entities allocated on the LPs should dynamically map the subset of entities having the highest rate of interactions during the current phase of the simulation. This dynamic mechanism will be exploited to realize adaptation-based clustering, that is, to capture the dense interactions (event-messages communication) without speculating on the density assumption of model entities, and by optimizing both load balancing and communication overheads. In conclusion, the load balancing mechanism is implemented, without assuming a static partition and allocation of the simulated area, and by maintaining the ''proximity'' clustering of model entities for the reduction of communication overheads. This would enable load balanced execution of scenarios characterized by non-uniform entity distributions and event-generation density, like in dynamic hot spots originated at simulation runtime, for which it was impossible to maintain load balancing in the MoVES I implementation. As an example, it is worth noting that Italian towns usually have concentric streets, variable street sizes and intersection density, due to the Middle Ages origins (see Fig. 6); hence hot-spot distribution of vehicles

may easily appear in quite an unpredictable way, caused by traffic jams.

4.5.1. The ARTIS communication overhead reduction (COR) heuristic

The Communication Overhead Reduction (COR) heuristic is based on a data structure associated with each model entity by the ARTIS middleware. A sliding window approach, whose size is a simulation parameter, is used to limit the time-horizon of the heuristic. To limit the overhead, the heuristic function is called only after a communication is performed, involving the model entity. After a time-step including at least a communication event, the heuristic evaluates the external LP which was destination of the majority of external model entity interactions, during the recent sliding window observation period. If the rate of the external (foreign-LP) with respect to internal (home-LP) interactions is greater than a threshold parameter (which can be tuned at runtime), then a migration of the model entity is performed towards the external-LP. This COR heuristic aims at maintaining a high Local Communication Ratio (LCR) in the communications between LPs, resulting in lightweight communication overheads.

4.5.2. The ARTIS load balancing (LB) heuristic

Combined with the COR heuristic, a Load Balancing (LB) heuristic is implemented by the ARTIS middleware. The LB heuristic monitors fast and slow LPs, respectively, when implementing a simulated time-step synchronization. A distributed voting-like procedure is performed to share the local ranking of each LP. In this way, based on a threshold parameter, an LP can realize if it can accommodate more load with respect to the average, or if it should deallocate some load. In both cases, the LP also identifies the target LP to migrate model entities to.

5. Testbed models for vehicular and wireless scenarios

In the following we illustrate the main characteristics of the vehicular and wireless communication models implemented in MoVES II, as a performance analysis testbed of the PADS architecture.

5.1. Vehicular model

The mobility- and area-entities in our model are road segments, lanes, intersections, traffic lights and the mobile vehicles. A Parser module can import maps from the GPS TrackMaker program [49] by creating and allocating data structures in memory according to selected map partition strategy. As proposed in STRAW [13], we identify the components of the road model as: an intra-segment component, an inter-segment component and a route management model. The intra-segment model controls vehicular movement into a road segment according to a bottom-up, microscopic approach. Several car-following models have been proposed in the literature to model the cars' behavior on a single-lane or multi-lane street. For driver characterization, we implement the psycho-physical model defined in [15], which is based on different parameters (speed of the vehicle, speed of the vehicle ahead, safe distance, driver's mood). Lane-change policies are currently not supported, but can be easily included in the model. Intersection entities are responsible for controlling the flow of vehicles from/to connected segments. Inter-segment mobility is modeled by using a kind of admission control/reservation scheme: when the leading vehicle in a lane reaches a threshold distance from the stop-line, it requests permission to occupy the intersection. The intersection entity then replies based on the current state of correlated entities (traffic lights, directions of vehicles currently being in the intersection, etc.). If it is allowed to, the vehicle entity passes the intersection, otherwise, it stops and waits for a notification. The MoVES II Route Management component manages the road segments that a vehicle traverses during the simulation run. A Random Walk model is considered in the performance analysis reported in this paper: vehicles randomly select the next road at each intersection. Other synthetic mobility models, like random waypoint patterns and more realistic traffic-aware and shortest-path strategies could be easily be integrated, and will be considered in future work. The additional amount of computation required to implement such strategies has been synthetically shaped in the performance analysis section, to measure the relative impact of variable distributed computation load.

5.2. Wireless model

In this paper, a prototype definition of a VANET model is used to study the analysis potential of inter-vehicle communication among mobile nodes, and the performance impact of the composition of vehicular and wireless models on the PADS architecture. Only a subset of the modeling potential of

MoVES II is exploited in this work, and the completion of model libraries is an ongoing activity. In this paper, the IEEE 802.11 standard [50] is considered as the enabling wireless technology for VANETs. A mobile wireless device defines a protocol-stack including the application layer, the data link and MAC layers. Two wireless signal propagation models are currently available: open-space and two-ray ground [35]. Some works on intelligent transportation systems (ITS) analysis consider the real-time dissemination of traffic information as the target application for VANETs. As an example, the SOTIS and the TrafficView projects [11,34] propose scalable traffic information systems that work by allowing a vehicle to periodically send and receive traffic information from neighbor vehicles. This information is shown on-board on display systems for augmented safety and driver assistance.

In our MoVES model, at the application layer, we consider two different approaches for data dissemination in vehicular systems [51]. In the first method, a Data Diffusion Protocol (DDP) is modeled as a constant flow of ping messages, as follows: each node (i) maintains an environment representation (ER) of the surroundings, (ii) updates the ER upon reception of new data, and (iii) periodically broadcasts the ER to the neighbor nodes. Another method for the dissemination of information is flooding: each node receiving a message immediately forwards it in its zone, using sequence-number lists and time-to-live mechanisms to limit useless retransmissions and network congestion. The MAC layer of each wireless device implements the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) Medium Access Control protocol of the IEEE 802.11 Distributed Coordination Function (DCF). A realistic simulation of inter-vehicle communication in urban scenarios would require realistic and accurate wireless propagation models, including the effects of obstacles like buildings, path-loss, multi-path, shadowing and diffractions effects. In the analysis shown in this work, a simple two-ray ground propagation model is used [35,9], since we are interested on the MoVES migration management, and on PADS performance effects, influenced by model factors like the local causality of broadcast communication, and the vehicle mobility and distribution.

6. Performance analysis

In this section, we present a preliminary illustration of MoVES performance. For space reasons, we only include a small subset of the results obtained from the simulation experiments, aiming to show different aspects of this work: (i) a general comparison of the MoVES I and the MoVES II implementations, from a performance viewpoint, (ii) the impact of the parallel/distributed approach on the micro-simulation of large-scale highly-populated urban traffic scenarios, (iii) benefits and drawbacks of adaptation mechanisms included in the PADS middleware.

A real-world urban area (the Middle Ages downtown of Bologna) is the target scenario (shown in Fig. 6): 1450 road segments and 760 intersections in a 5.75 km^2 area. Vehicle density is varied, in a range from 2000 to 16000 vehicles in the simulated area, both under uniform and hot-spot distributions. This is made in order to evaluate the effectiveness of sequential vs. parallel/distributed simulation approaches under different model assumptions and computation load.

We define both parallel and distributed simulation architectures for the analysis. The parallel simulation is implemented over one multiprocessor PEU with shared-memory CPUs. The distributed simulation is implemented over PEUs connected via a LAN. All the parallel simulation experiments have been executed on a Quadral Intel Pentium IV Xeon CPU 1.50 GHz, with Hyper-Threading support activated and 1 GB RAM (Quadral Parallel Architecture). All the distributed simulation experiments have been executed on a cluster of three homogeneous Dual Intel Pentium IV Xeon, 2.8 GHz, 3 GB RAM, interconnected by a Fast-Ethernet (100 Mb/s) LAN ($3 \times$ Dual Distributed Architecture). In all our experiments, each CPU executes one single LP. We performed multiple runs of each experiment, and the confidence intervals obtained with a 90% confidence level are lower than 5% the average value of the performance indexes shown.

6.1. MoVES I vs. MoVES II implementations

In this section, we show a performance comparison of the MoVES I and MoVES II implementations. The PADS execution is parallel, with 4 LPs implemented on the Quadral Parallel Architecture. The Wall Clock Time (WCT) is used as the performance index, because speedup is not a correct index to compare different models/implementations. In Fig. 7, we show the WCT needed to complete a testbed simulation run in a scenario with vehicles uni-

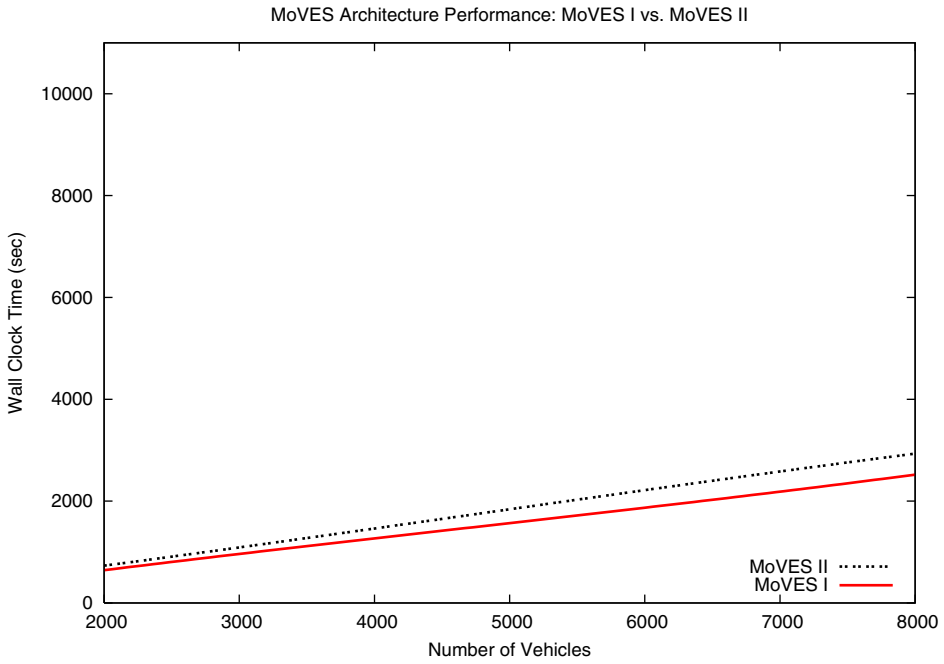

Fig. 7. MoVES I vs. MoVES II, uniform vehicle distribution, parallel simulation with 4 LPs (4 CPUs).

formly distributed in the simulated area. The MoVES I framework design is based on uniform distribution assumption, hence MoVES I is working under its most favorable conditions in this experiment. Vehicles are mobile and wireless-enabled (IEEE 802.11 technology). In addition, vehicles are performing significant computation due to the implementation of the Data Diffusion Protocol (DDP) described in Section 5.2. In all the tests, to support the detailed modeling of wireless protocols, the time-step was fixed to 10^{-5} s of simulated time. The complex migration management and state sharing communication needed by the MoVES II implementation is expected to introduce overheads with respect to the MOVES I implementation. Fig. 7 shows that a small overhead is introduced by MoVES II, but the impact of the overhead is under control for the considered scenario. Fig. 8 shows the same performance index, obtained under a different scenario: in the initialization phase, all vehicles are uniformly distributed in the area, and the mobility model of vehicles follows a non-uniform distribution. All vehicles start moving to a small region of the simulated area (hot-spot distribution). In this case, MoVES I cannot react to computation-load unbalancing, because a majority of vehicles will be eventually clustered in a small region of the simulated area. This means that a subset of LPs will be managing high computation loads. This effect has a significant impact on the MoVES I simulation performance, as shown in Fig. 8. The MoVES II implementation is faster than MoVES I, in terms of simulation WCT. The same qualitative results, not shown for space reasons, have been obtained under distributed execution architectures.

Results shown in Figs. 7 and 8 demonstrate that the MoVES II implementation is scalable and adapts in an automatic fashion to realistic model dynamics, with low-overheads and significant performance gain, with respect to the MOVES I implementation. For this reason, in the following we will concentrate our analysis on the MoVES II implementation, only.

6.2. Analysis of mobile vehicular scenarios

In the following we have tested the scalability of the MoVES II framework, in parallel and in distributed execution environments, when only the vehicular-mobility is simulated (that is, without considering the computation and synchronization effect introduced in the simulation by wireless propagation, communications and the DDP application). In all tests, the time-step was fixed to 1 s of simulated time, while the number of LPs involved in a simulation run is varied between 1 and 6 (in the distributed scenario), and between 1 and 4 (in

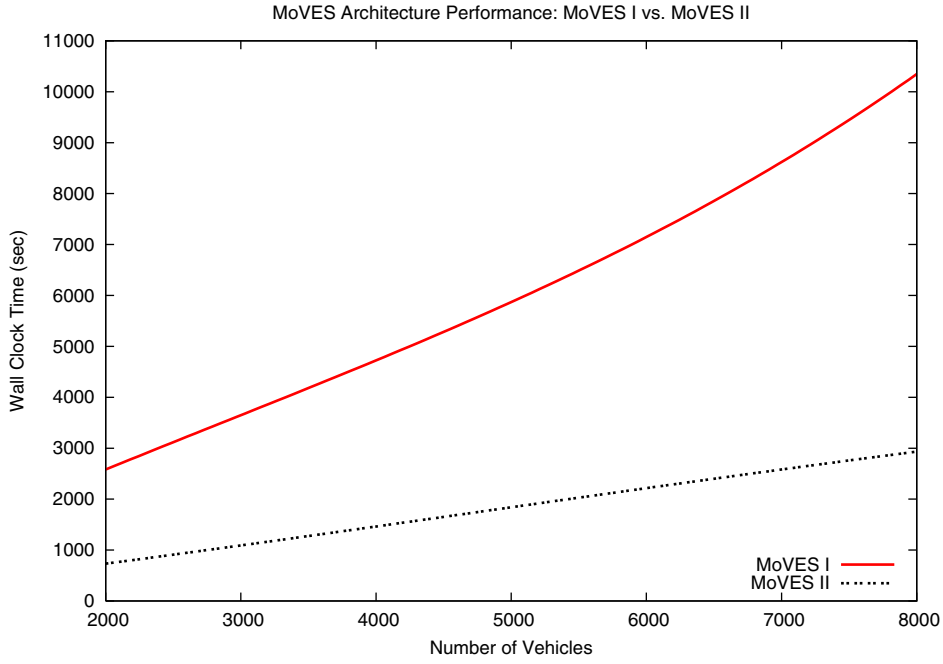

Fig. 8. MoVES I vs. MoVES II, hot-spot vehicle distribution, parallel simulation with 4 LPs (4 CPUs).

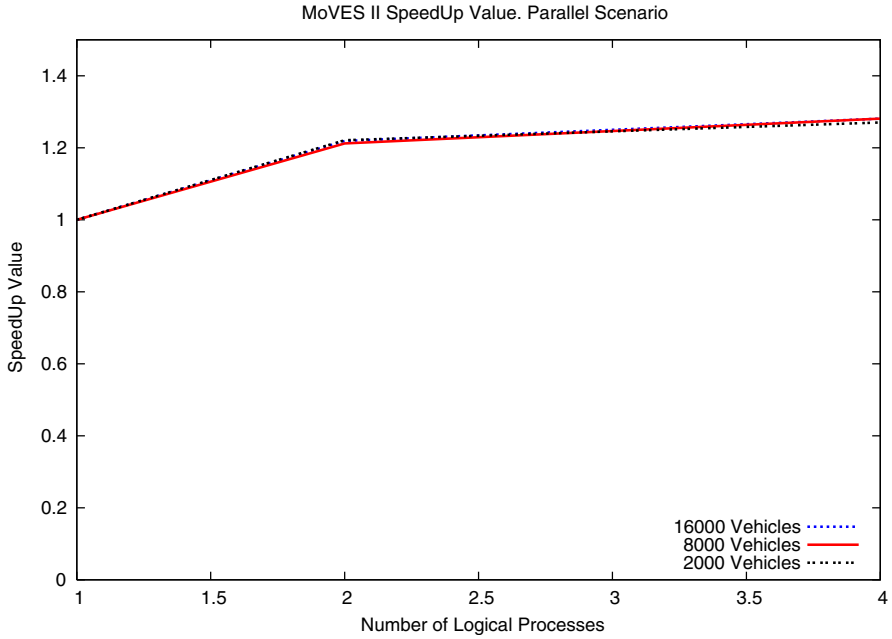

Fig. 9. Speedup value, parallel execution, variable number of mobile vehicles.

the parallel scenario). The scalability evaluation is limited to 4 and 6 LPs, respectively, due to the current availability of dedicated and homogeneous resources in our execution platform. This scalability analysis will be possibly extended as a future work. For a given simulation run, the speedup is defined as the ratio of execution time of a monolithic simulation, divided by the execution time of parallel/distributed simulation [20]. This is considered the main index to evaluate the performance

gain resulting from parallel/distributed computation. The speedup is affected by different factors, including load balancing, memory utilization, synchronization and message passing overheads required for the coordination of LPs [20]. Fig. 9 shows the speedup value for a parallel execution (on the Quadral Parallel Architecture), as a function of the number of LPs. The figure shows that a small speedup can be obtained by increasing the number of LPs. The difference between the scenarios with

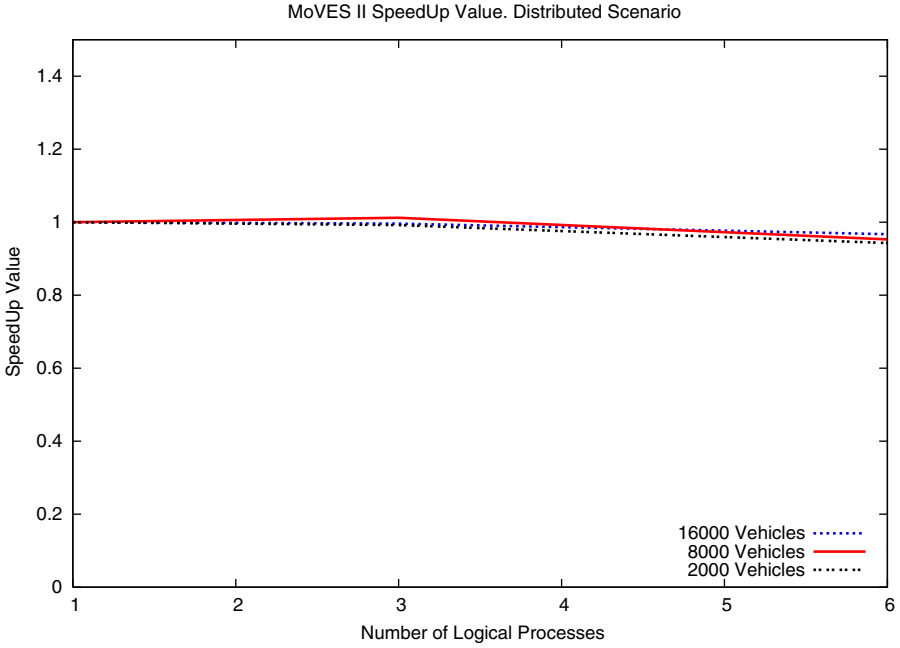

Fig. 10. Speedup value, distributed execution, variable number of mobile vehicles.

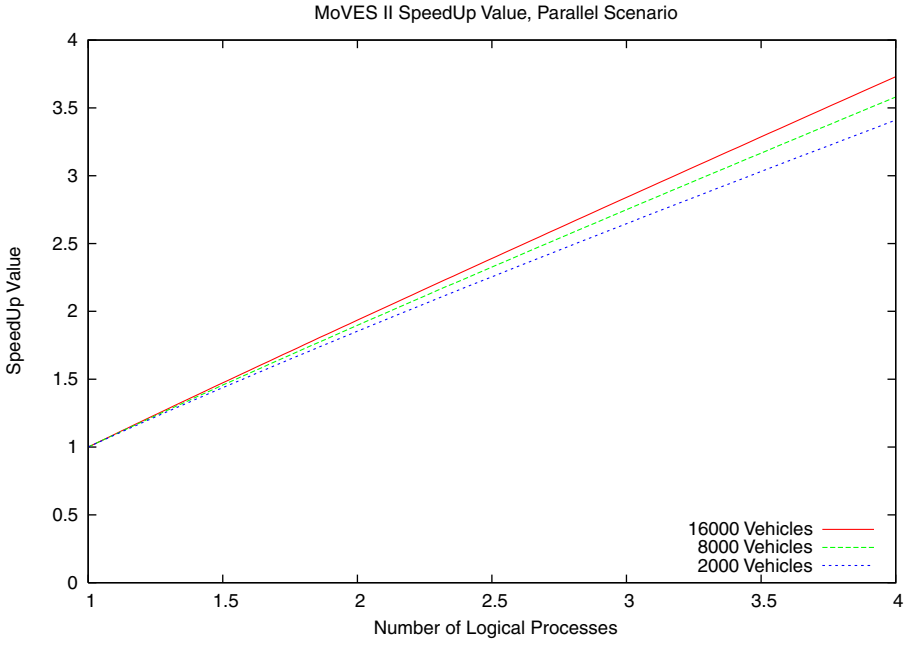

Fig. 11. Speedup value, parallel execution, variable number of mobile vehicles, high computation model.

2000, 8000 and 16000 vehicles is not significant. This demonstrates that the computation load in this model is marginal, resulting in a low advantage of the parallel execution.

Fig. 10 shows the speedup value of a distributed execution (on the $3 \times$ Dual Distributed Architecture), as a function of the number of LPs. In this distributed scenario, no speedup is obtained for the considered simulated model, and the differences obtained by varying the number of simulated vehicles are marginal. No speedup is obtained in this case because the modeled scenario has a high degree of communication overheads (to maintain a shared state and to maintain synchronization) associated with a low degree of computation. The parallel implementation in Fig. 9 has slightly better speedup than the distributed implementation, because the cost of communication (and synchronization) is affected by the latency of a LAN, which is greater than the latency of a shared memory communication. For this scenario, a similar performance is obtained under a monolithic and a PADS simulation. Beyond performance results, the scalability resulting from resources aggregation in a PADS simulation for the latter scenario would be useful (i) to support a huge number of vehicles in a wide simulation area, or (ii) to support computation intensive protocol stacks and user applications on the modeled entities.

The potential speedup that can be obtained by exploiting concurrent computation in a PADS has been tested by increasing the computation load in the previous model, as shown in Figs. 11 and 12. Each vehicle now performs "high computation" by calculating an adaptive-path between its current location and a target destination, which requires the frequent execution of a modified version of the Dijkstra algorithm on a weighted street graph. In other words, this has been defined as a simulation testbed aiming to stress the computation load in the system. Fig. 11 shows the speedup of a parallel execution of the "high computation" model: the speedup obtained approximates the maximum parallelism achievable with up to four parallel LPs. Fig. 12 shows the speedup index obtained for a distributed execution of the "high computation" model in a range up to six distributed LPs: again, the speedup obtained approximates the maximum parallelism achievable for the execution architecture considered. In this case, the speedup is smaller than the one obtained in the parallel execution due to the overhead of communication in distributed (LAN) environment, which is higher than shared memory communication. In Figs. 11 and 12 the number of simulated vehicles has a remarkable effect on the speedup, due to the multiplication of the high per-vehicle model computation introduced. In Figs. 9 and 10 the effect of the variable number of vehicles

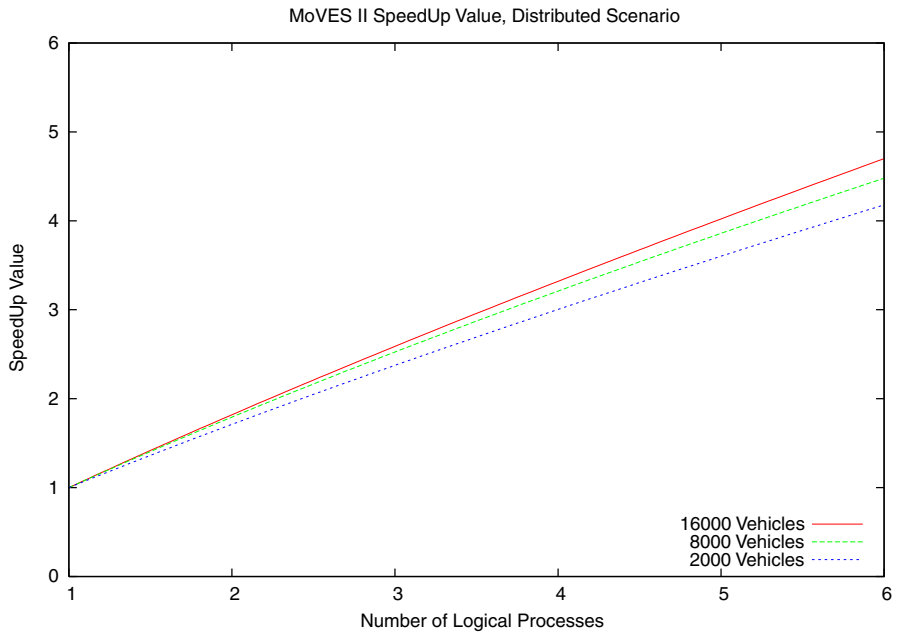

Fig. 12. Speedup value, distributed execution, variable number of mobile vehicles, high computation model.

is negligible due to the low model computation required.

The speedup obtained in Figs. 11 and 12 is based on high computation assumption. In the following we will show a testbed analysis of a model including both vehicular-mobility and IEEE 802.11 wireless communication, that is, a more natural model for the MoVES II design. Since the implementation of the detailed simulation of wireless protocols requires a computation intensive activity, we expect that this simulated scenario would obtain typical speedup results in the range of the results obtained with previous model assumptions.

6.3. Analysis under mobile vehicular and wireless scenarios

In this section we test the scalability of the MoVES framework in the simulation of VANETs, that is, when both the mobility factor and the wireless inter-vehicle communication are modeled. Inter-vehicle communication is performed for the implementation of the Data Diffusion Protocol (DDP) described in Section 5.2. Other modeling parameters of wireless communication appear in Table 1. The time-step of the simulation is determined by the smallest time parameter of the IEEE 802.11 MAC protocol [50], that is, a SIFS Time (10 μs): this means a 10^5 reduction of the time-step size with respect to pure mobility simulation scenarios. Since it is reasonable that in future vehicular ad hoc networks only a fraction of vehicles will be equipped with a wireless interface, we introduce a penetration rate factor (PR) representing the fraction of vehicles which have a wireless device.

Table 1
Wireless model parameters

Simulation time step	10 μs (IEEE 802.11 SIFS)
MAC layer	IEEE 802.11 DCF
PR factor (%wireless vehicles)	50%
Packet size	64 Byte
Packet rate	1 pkt/s
Propagation model	
(Avg. Transmission range)	Two-Ray Ground
(Avg. Carrier Sense range)	100 m
	300 m
Vehicle Density	1 vehicle × 3125 m^2
Nominal Channel Bitrate	2 Mbps

Fig. 13 shows the speedup value when the number of vehicles varies between 2000 and 16000, and the wireless penetration rate is 50%. In order to have a correct comparison in terms of scalability, the scenarios with 2000, 8000 and 16000 vehicles have been defined such that the vehicle density in the simulated area is constant, and vehicles are uniformly distributed. The execution architecture is a distributed

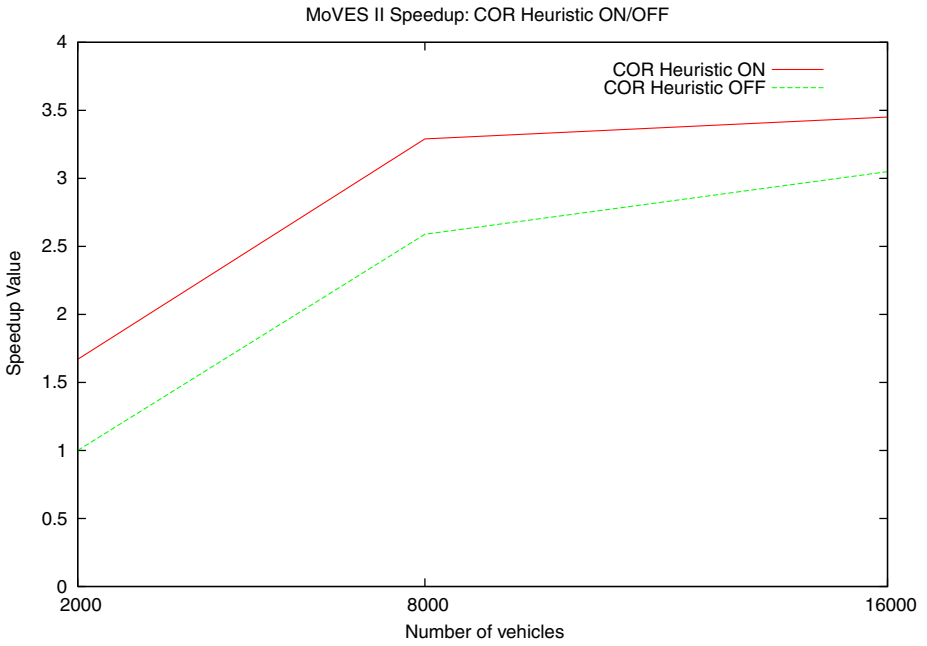

Fig. 13. Distributed simulation, six LPs, VANET scenario, communication overhead reduction (COR) ON/OFF.

simulation with six LPs, executed on a 3 × Dual Distributed Architecture. In general, parallel architectures have the best performance results for the scenario considered. Since the distributed architecture is the most critical and interesting for simulation, under the communication overheads viewpoint, we will focus on distributed implementation results. By looking at Fig. 13, we observe that the speedup obtained by the distributed implementation when the MoVES II Communication Reduction heuristic is disabled (COR OFF) is 1.0 (that is, equivalent to the monolithic approach) with 2000 vehicles, 2.59 with 8000 vehicles, and 3.05 with 16000 vehicles. This confirms that a computation intensive activity, like the wireless protocol implementation in the vehicular scenario, could obtain significant speedup results. Since the distributed scenario is sensitive to the cost of communication (and synchronization), which is influenced by high latency of LANs, a communication overhead reduction heuristic (COR) like the one defined in Section 4.5.1 would give additional advantages. This is demonstrated in Fig. 13, by looking at the speedup values obtained in the vehicular wireless scenario with MoVES II and the COR heuristic activated (COR ON): 1.67 with 2000 vehicles, 3.29 with 8000 vehicles, and 3.45 with 16000 vehicles. The speedup gain due to the COR heuristic can be obtained as the difference between the two values, resulting in 0.67, 0.70 and 0.40 for 2000, 8000 and 16000 vehicles, respectively.

Fig. 14 shows the average LCR index (see Section 4.3.3) for the six LPs realizing the distributed simulation of the vehicular wireless scenario. This figure demonstrates the effectiveness of the COR heuristic in the reduction of the communication overhead, obtained with dynamic, migration-based clustering of mobile model entities. The COR-based migration scheme is devoted to maintain an elevated Local Communication Ratio (LCR) in the communication between LPs, that is, lightweight communication overheads. Fig. 14 shows the static LCR value obtained between six LPs without the effect of the COR heuristic. This is equivalent to the probability that a local entity allocated on LP X randomly selects a destination entity located on the same LP. By assuming that the allocation of model entities on six LPs follows a uniform distribution, the expected value is $LCR = 1/6$. Given the initial random allocation of model entities, the migration-based overhead communication reduction implemented by the COR heuristic is able to dynamically cluster many interacting entities on the same LPs. As shown in Fig. 14, the steady state percentage of LCR with COR heuristic activated reaches values between 75% and 80% after a short initial transient phase.

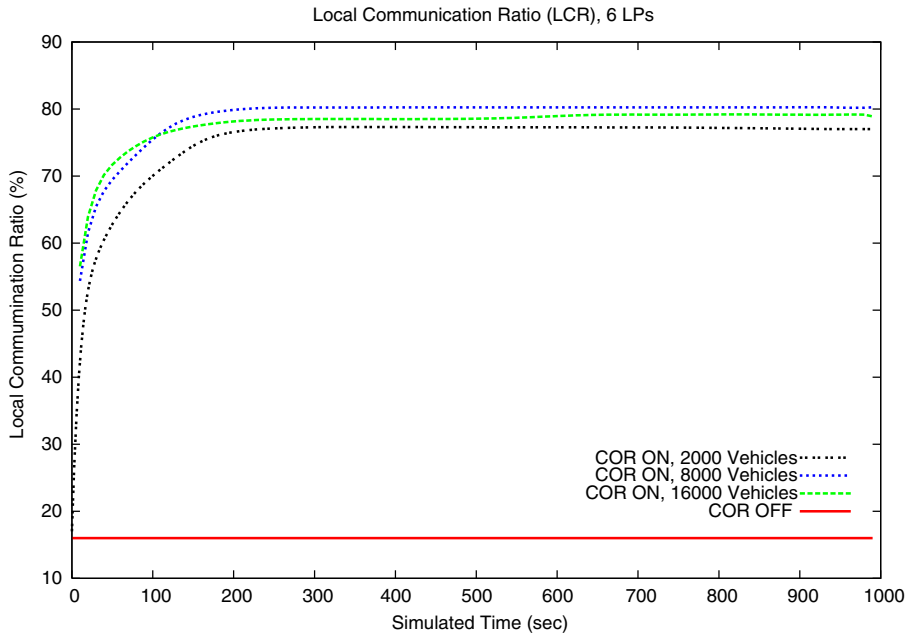

Fig. 14. Average LCR of six distributed LPs, vehicular wireless scenario, COR heuristic ON/OFF.

7. Conclusions and future work

In this paper, we have illustrated the architecture and design issues of a novel, scalable and efficient framework for the parallel and distributed simulation of vehicular ad hoc networks, named Mobile Wireless Vehicular Environment Simulation (MoVES). MoVES is implemented on top of the ARTIS simulation middleware, by including solutions for communication overhead reduction, and for computation- and communication-load balancing based on model-entity migration heuristics for parallel and distributed simulation. The proposed solutions have shown effective performance improvements in the simulation of realistic wireless vehicular scenarios. We have defined and tested vehicle mobility in a real urban area model, including driver psycho-physical model, inter-vehicle wireless communication, signal propagation, IEEE 802.11 MAC (DCF) protocol, and traffic dissemination applications. The performance analysis has demonstrated the scalability, efficiency and accuracy of MoVES, and some guidelines about the possible drawbacks and benefits of a parallel/distributed simulation of communication solutions for mobile wireless VANETs.

Future work includes the improvements of the ARTIS middleware support, and MoVES framework model libraries. In addition, we will investigate solutions to increase the adaptation of the simulation framework to the characteristics of the model and the execution architecture. The new solutions will be designed to increase the performance and scalability of the PADS simulation framework and the support for layered and module-based modeling and simulation of dynamic and massive wireless vehicular, sensor and mesh networks.

Acknowledgements

This work is partially supported by Italian MIUR funds under the PRIN project NADIR (Design and performance evaluation of protocols and distributed algorithms for quality of service in mesh networks) and by University of Bologna funds, under the project ESOV (Enabling Services on Vehicles).

Authors wish to thank Prof. Nael Abu-Ghazaleh and all the anonymous reviewers for their suggestions that significantly helped to improve the paper clarity and contribution.

References

[1] J. Blum, A. Eskandarian, L. Hoffman, Challenges of intervehicle ad hoc networks, IEEE Transactions on Intelligent Transportation Systems 5 (4) (2004) 347–351.
[2] Cooperative Vehicle Infrastructure System (CVIS) Integrated Project, European Commission, Sixth Framework Programme of Research, 02/2006-01/2010. <http://www.cvisproject.org/>.
[3] M. Nekovee, Sensor networks on the road: the promises and challenges of vehicular adhoc networks and vehicular grids, in: Proceedings of Workshop on Ubiquitous Computing and e-Research, Edinburgh, UK, May 2005.
[4] IEEE 802.11 TGp, Wireless Access for Vehicular Environments (WAVE).
[5] CALM, Continuous Air interface for Long and Medium range, ETSI, ISO TC204 WG16.
[6] IEEE Std 1516-2000: IEEE Standard for modelling and simulation (M&S) high level architecture (HLA).
[7] W. Franz, R. Eberhardt, T. Luckenbach, FleetNet- Internet on the road, in: Proceedings of ITS'01, Sydney, Australia, October 2001.
[8] Network on Wheels Project. <http://www.prevent-ip.org>.
[9] D. Reichardt, M. Miglietta, L. Moretti, P. Morsink, W. Schulz, CarTalk 2000: safe and comfortable driving based upon inter-vehicle communication, in: Proceedings of IVS'02, Versailles, France, June 2002, pp. 545–550.
[10] M. Schulze, G. Nocker, K. Bohm, PReVENT: a European program to improve active safety, in: Proceedings of International Conference on ITS Telecom., Brest, France, June 2005.
[11] T. Nadeem, S. Dashtinezhad, C. Liao, L. Iftode, TrafficView: traffic data dissemination using car-to-car communication, ACM Sigmobile Mobile Computing and Communications Review 8 (3) (2004) 6–19.
[12] L. Wischhof, A. Ebner, M. Lott, H. Rohling, R. Halfmann, SOTIS: a self-organizing traffic information system, in: Proceedings of VTC'03, Jeju, South Korea, April 2003, pp. 2242–2246.
[13] D. Choffnes, F.E. Bustamante, STRAW: an integrated mobility and traffic model for VANETs, in: Proceedings of CCRTS'05, Mclean, VA, USA, June 2005.
[14] M. Moske, H. Fussler, H. Hartenstein, W. Franz, Performance measurements of a vehicular ad hoc network, in: Proceedings of VTC'04, Los Angeles, CA, USA, September 2004, pp. 2116–2120.
[15] S. Panwai, H. Dia, Comparative evaluation of microscopic car-following behavior, IEEE Transactions on Intelligent Transportation Systems 6 (3) (2005) 314–325.
[16] T.S. Rappaport, Wireless Communications: Principles and Practice, Prentice Hall, 2002.
[17] R. Klefstad, Y. Zhang, M. Lai, R. Jayakrishnan, R. Lavanya, A scalable, synchronized, and distributed framework for large-scale microscopic traffic simulation, in: Proceedings of ITSC'05, Vienna, Austria, September 2005.
[18] H. Liu, W. Ma, R. Jayakrishnan, W. Recker, Distributed large-scale network modeling with paramics implementation, in: Proceedings of ITS'05, Vienna, Austria, September 2005, pp. 232–238.
[19] G.D.B. Cameron, G.I.D. Duncan, PARAMICS: parallel microscopic simulation of road traffic, Springer Journal of Supercomputing 10 (1) (1996) 25–53.

[20] R.M. Fujimoto, Parallel and Distributed Simulation Systems, Wiley-Interscience, 1999.
[21] U. Klein, T. Schulze, S. Strassburger, Traffic simulation based on the high level architecture, in: Proceedings of Proc. of WSC'98, Washington, D.C., USA, December 1998, pp. 1095–1104.
[22] L. Bononi, M. Di Felice, M. Bertini, E. Croci, Parallel and distributed simulation of wireless vehicular ad hoc networks, in: Proceedings of MSWiM'06, Torremolinos, Spain, October 2006, pp. 28–35.
[23] L. Bononi, M. Bracuto, G. D'Angelo, L. Donatiello, Performance analysis of a parallel and distributed framework for large scale wireless systems' simulation, in: Proceedings of MSWiM'04, Venice, Italy, October 2004, pp. 52–61.
[24] L. Bononi, M. Bracuto, G. D'Angelo, L. Donatiello, ARTIS: a parallel and distributed simulation middleware for performance evaluation, in: Proceedings of ISCIS'04, Kemer-Antalya, Turkey, October 2004, pp. 627–637.
[25] PADS research group, Department of computer science, University of Bologna, Italy, The ARTIS documentation and distribution webpage. <http://pads.cs.unibo.it/>.
[26] VISSIM PTV Simulation Tool. <http://www.english.ptv.de/cgi-bin/traffic/traf_vissim.pl>.
[27] Quadstone PARAMICS, Paramics suite of traffic simulation tools. <http://www.paramics-online.com/>.
[28] Traffic Group inc. SimTraffic 7.0 (Traffic Engineering and Transportation Planning). <http://www.trafficgroup.com/>.
[29] C. Schroth, F. Dötzer, T. Kosch, B. Ostermaier, M. Strassberger, Simulating the traffic effects of vehicle-to-vehicle messaging systems, in: Proceedings of International Conference on ITS Telecom., Brest, France, June 2005.
[30] H. Wu, J. Lee, M. Hunter, R. Fujimoto, R.L. Guensler, J. Ko, Efficiency of simulated vehicle-to-vehicle message propagation in Atlanta, Georgia, I75 corridor, Journal of the Transportation Research Board 1910 (2005) 82–89.
[31] Scalable Network Technologies, The first natively parallel network simulator, Qualnet 4.0. <http://www.scalable-networks.com/>.
[32] JiST/SWANS. <http://jist.ece.cornell.edu/>.
[33] A.K. Saha, D.B. Johnson, Modeling mobility for vehicular ad-hoc networks, in: Proceedings of VANET'04, Philadelphia, PA, USA, October 2004, pp. 91–92.
[34] D.-H. Lee, P. Chandrasekar, A framework for parallel traffic simulation using multiple instancing of a simulation program, Intelligent Transportation Systems 7 (3–4) (2002) 279–294.
[35] A. Boukerche, L. Bononi, Simulation and modeling of wireless, mobile and ad hoc networks, in: S. Basagni, M. Conti, S. Giordano, I. Stojmenovic (Eds.), Mobile Ad Hoc Networking, IEEE/Wiley, 2004.
[36] The Network Simulator, ns-2. <http://nsnam.isi.edu/nsnam/>.
[37] Omnet++ Community Site. <http://www.omnetpp.org/>.
[38] Opnet Technologies. <http://www.opnet.com/>.
[39] R.L. Bagrodia, W.T. Liao, Maisie: a language for the design of efficient discrete-event simulations, IEEE Transactions on Software Engineering 20 (4) (1994) 225–238.
[40] Parallel/ Distributed ns. <http://www.cc.gatech.edu/computing/compass/pdns/>.
[41] GTNetS (Georgia Tech Network Simulator). <http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/>.
[42] Dartmouth SSF (DaSSF). <http://www.cs.dartmouth.edu/research/DaSSF/>.
[43] iSSF Home Page. <http://www.crhc.uiuc.edu/~jasonliu/projects/issf/>.
[44] GTW/TeD/PNNI. <http://www.cc.gatech.edu/computing/pads/teddoc.html>.
[45] O.E. Kelly, J. Lai, N.B. Mandayam, A.T. Ogielski, J. Panchal, R.D. Yates, Parallel simulations of wireless networks with TED: radio propagation, mobility and protocols, SIGMETRICS 25 (4) (1998) 30–39.
[46] A. Boukerche, S.K. Das, A. Fabbri, SWiMNet: a scalable parallel simulation testbed for wireless and mobile networks, Wireless Networks 7 (5) (2001) 467–486.
[47] R.L. Bagrodia, X. Zeng, M. Gerla, GloMoSim: a library for parallel simulation of large-scale wireless networks, in: Proceedings of PADS'98, Banff, Canada, May 1998, pp. 154–161.
[48] S. Eichler, B. Ostermaier, C. Schroth, T. Kosch, Simulation of car-to-car messaging: analyzing the impact on road traffic, in: Proceedings of MASCOTS'05, Atlanta, GA, USA, September 2005, pp. 507–510.
[49] GPSTrackMaker Homepage. <http://gpstm.cpm/>.
[50] IEEE 802.11 WG, IEEE Std. 802.11, 1999 ed, Part II: Wireless LAN MAC and PHY layer specs. 1999.
[51] J. Bronsted, L. Krinstensen, Specification and performance evaluation of two zone dissemination protocols for vehicular ad hoc networks, in: Proceedings of ANSS'06, Huntsville, AL, USA, April 2006, pp. 68–79.

Luciano Bononi received the Laurea degree (Summa Cum Laude) in Computer Science in 1997, and the PhD in Computer Science in 2002, both from the University of Bologna, Italy. In spring 2000 he was a visiting researcher at the Department of Electrical Engineering of the University of California at Los Angeles. From March 2001 to September 2002 he was a postdoc researcher, and since October 2002 he is an assistant professor at the Department of Computer Science of the University of Bologna. His research interests include mobile and wireless network protocols, standards and architectures, QoS and security, network on chip architectures, communication protocol design and analysis, modeling and simulation, parallel and distributed simulation.

Marco Di Felice received the Laurea degree (summa cum laude) in Computer Science in 2004, from the University of Bologna, Italy. Currently, he is a PhD student in the Department of Computer Science at the University of Bologna, Italy. His research activity includes mobile networking and computing, design, analysis and performance evaluation of protocols and architectures for wireless ad hoc networks, vehicular networks, modeling and simulation of wireless systems.

Gabriele D'Angelo received the Laurea degree (summa cum laude) in Computer Science in 2001, and a PhD degree in Computer Science in 2005, both from the University of Bologna, Italy. He is an Assistant Professor at the Department of Computer Science of the University of Bologna. His research interests include parallel and distributed simulation, distributed systems, Internet games and computer security. He is the author of several publications on these topics. During the last few years he has worked on the design and implementation of the ARTIS parallel and distributed simulation middleware.

Michele Bracuto received the Laurea degree in Computer Science in 2002, from the University of Bologna, Italy. He is a research associate at the Department of Computer Science of the University of Bologna, Italy. His research activity includes parallel and distributed simulation, distributed systems and Quality-Of-Service of multimedia on Internet. During the last few years he worked on the design and implementation of the ARTIS parallel and distributed simulation middleware.

Lorenzo Donatiello received the Laurea degree in computer science from University of Pisa. From 1984 to 1990, he was with the Department of Computer Science, University of Pisa. Since November 1990, he has been a professor of Computer Science at the University of Bologna. From March 1983 to April 1984 and from August 1986 to October 1986, he was a visiting scientist at the IBM T.J. Watson Research Center, Yorktown Heights, NY, USA. His research interests include performance models of computer and communication systems, software architecture, parallel and distributed simulation, wireless networks.