GloMoSim: A Library for Parallel Simulation of Large-scale Wireless Networks

Xiang Zeng Rajive Bagrodia Mario Gerla
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA 90095

Abstract

A number of library-based parallel and sequential network simulators have been designed. This paper describes a library, called GloMoSim (for Global Mobile system Simulator), for parallel simulation of wireless networks. GloMoSim has been designed to be extensible and composable: the communication protocol stack for wireless networks is divided into a set of layers, each with its own API. Models of protocols at one layer interact with those at a lower (or higher) layer only via these APIs. The modular implementation enables consistent comparison of multiple protocols at a given layer. The parallel implementation of GloMoSim can be executed using a variety of conservative synchronization protocols, which include the null message and conditional event algorithms. This paper describes the GloMoSim library, addresses a number of issues relevant to its parallelization, and presents a set of experimental results on the IBM 9076 SP, a distributed memory multi-computer. These experiments use models constructed from the library modules.

1 Introduction

The rapid advancement in portable computing platforms and wireless communication technology has led to significant interest in mobile computing and mobile networking. Two primary forms of mobile computing are becoming popular: first, mobile computers continue to heavily use wired network infrastructures. Instead of being hardwired to a single location (or IP address), a computer can dynamically move to multiple locations while maintaining application transparency. Protocols such as Mobile-IP have been suggested to handle message routing in this scenario. This is somewhat analogous to cellular telephony, where mobile subscribers must always communicate over a single hop to a base station, which subsequently routes the call over a fixed network. In the second form, called *ad hoc networking*, communication among a set of mobile units is established 'on the fly' without using any existing network infrastructure. In its most general form, ad hoc networking supports transmission of multimedia traffic over multi-hop transmissions in a network of mobile communicating devices [1]. It is fairly easy to devise mobile networking scenarios where a mobile multi-hop wireless network is interfaced with a wired network such that end-to-end performance is determined by the characteristics of both wireless and wired network components.

Protocols designed for this kind of environment are complex to evaluate analytically due to factors such as complex channel access protocols, node mobility, channel propagation properties, and radio characteristics. Excessive execution time of detailed models forms a barrier for the effective use of simulation. In this paper we explore the use of parallel discrete event simulation to reduce execution time for composable detailed simulation model of wireless networks.

GloMoSim is a *library-based* sequential and *parallel* simulator for wireless networks. It is designed as a set of library modules, each of which simulates a specific wireless communication protocol in the protocol stack. The library has been developed using PARSEC, a C-based parallel simulation language [4]. New protocols and modules can be programmed and added to the library using this language. GloMoSim has been designed to be extensible and composable. It has been implemented on both shared memory and distributed memory computers and can be executed using a variety of synchronization protocols. In this paper we will focus on parallel performance study on a distributed memory multi-computer IBM 9076 SP.

The remainder of this paper is organized as follows: section 2 discusses related work in the area of network simulation. Section 3 outlines the wireless system architecture that is considered in this paper. Section 4

describes our primary simulation design approach. Section 5 discusses a number of issues that are relevant to parallel simulation of large-scale mobile networks. Section 6 presents some experimental results for parallel performance study and section 7 is the conclusion.

2 Related work

Much of the existing research in *parallel* network simulation has been done in the context of wired networks that include ATM networks, LAN simulators, and interconnection networks for parallel computers [3,7,9,14]. Relatively little research has been done to evaluate the feasibility of parallel simulation of wireless networks. Wireless and wired networks differ in fundamental ways: for example, the signal interference and attenuation concerns are inherently more complicated and typically more computationally intensive for wireless mediums than for wired mediums. Also, the broadcast nature of wireless radio transmission makes communication topology in simulation models relatively denser than for an equivalent wired network.

The research described in [12] studied the performance of parallel simulation of a specific protocol – a clustering algorithm in mobile, multi-hop, wireless networks. This paper attempts to develop an entire set of protocols by designing a library containing parallelized modules that can simply be composed together to evaluate the performance of different implementations of protocol stacks for *very large networks*.

Another closely related effort is the parallel simulation of PCS networks [5]. However, there are significant differences both in network characteristics and simulation research issues. First, PCS is a cell-centralized communication system, where each node can only talk with the base station assigned to the cell. Wireless networks considered in this paper include multi-hop networks. Second, the wireless communication medium can be modeled with differing detail that involves different complexity in parallel simulation. Third, in PCS models, communication is limited within a cell except that mobility may cause inter-cell and hence, inter-processor communication in parallel execution. In parallel simulation of a wireless network, both node mobility and ordinary communication may cause inter-processor communication (because signal interference among neighboring cells' propagation is explicitly modeled), which dramatically increases inter-processor communication topology.

3 Network architecture

In this paper we focus on the simulation of ad hoc networks. The networking stack is decomposed into a number of layers as shown in Figure 1. A number of protocols have been developed at each layer and models of these protocols or layers can be developed at different levels of granularity.

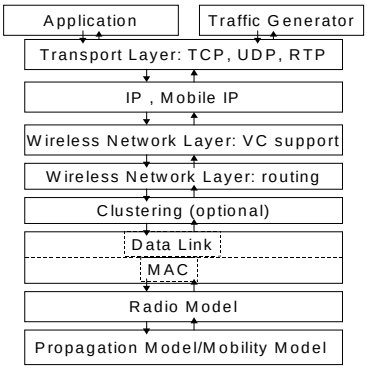

Figure 1. GloMoSim Architecture

For example, the channel propagation layer includes a *free space* model that calculates signal strength based only on the distance between every source and receiver pair; an *analytical* model that computes signal attenuation using a log normal distribution; a *fading* channel model that is computationally much more expensive but incorporates the effect of multi-path, shadowing and fading in calculating signal strength. As an example of alternative protocols, consider the Data Link/MAC layer, where a number of protocols have been suggested including: carrier sense multiple access (CSMA), multiple access collision avoidance (MACA [11]) and floor acquisition multiple access (FAMA). Each of these has been modeled in the GloMoSim library. At the network layer, flooding protocol, flat distance vector routing protocol DSDV is also contained in the library. We are still developing hierarchical routing protocol to handle scaled network routing. At transport layer, a TCP/IP simulator based on Free BSD 2.2.2 implementation has been built into library.

A common API between every two neighboring models on protocol stacks is predefined to support their composition. These APIs specify parameter exchanges and services between neighboring layers. For example, interfaces between Data Link/MAC layer and network layer are defined as message passing with following formats in the simulation library:

Packet Handling APIs:

Packet_from_NW_to_DLC(P_type, P_dest, P_source, P_payload, P_size, P_VCID)

Packet_from_DLC_to_NW(P_type, P_dest, P_source, P_payload, P_size, P_VCID)

P_type refers to the type of packet (data packets, control packets, etc.), *P_dest* and *P_source* refer respectively to source and destination node, and the other parameters are required for packet processing or quality of service support. Each protocol module at a given layer is required to comply with the APIs defined for that layer.

4 Simulation design: wireless model library

GloMoSim aims to develop a modular simulation environment for protocol stacks described in the previous section that is capable of scaling up to networks with thousands of heterogeneous nodes. If all protocol models obey the strict APIs defined at each layer, it will be feasible to simply swap protocol models at a certain layer (say evaluate the impact of using CSMA rather than MACA as the media access control protocol) without having to modify the models for the remaining layers in the stack.

GloMoSim library is written in PARSEC (for PARallel Simulation Environment for Complex Systems) [4], a simulation environment derived from the Maisie simulator [2]. PARSEC adopts a message-based approach to discrete-event simulation: physical processes are modeled by simulation objects called entities, and events are represented by transmission of time-stamped messages among corresponding entities. A visual-programming environment called PAVE (for PARSEC Visual Environment) has also been developed to support the visual design of PARSEC programs or to visually configure simulation models using pre-defined components for a library in a specific application domain like GloMoSim.

The requirements of scalability and modularity make the library design a challenging issue. A straightforward approach is to map each network node to a single simulation object, i.e. a PARSEC entity instance. However, previous experience has indicated that a large number of simulation objects can considerably increase simulation overhead, and such design is not scalable. For example, in order to simulate networks with more than 100,000 mobile nodes, at least 100,000 entity instances have to be created. This was found to be untenable as well as impractical.

Instead, GloMoSim assumes that the network is decomposed into a number of partitions and a single entity is defined to simulate a single layer of the complete protocol stack for all the network nodes that belong to the partition. Interactions among the entities must obey the corresponding APIs described in the previous section. Syntactically, the interactions may be specified using messages, function calls, or entity parameters as appropriate. This method supports modularity because a PARSEC library entity representing a layer of the protocol stack is largely self-contained. It encapsulates the complexity of a specific network behavior independently from other ones. This method also supports scalability because node aggregation inside one entity will be able to reduce the total number of entities which has been found to improve sequential performance even more dramatically than parallel performance.

A number of protocols have already been implemented into the library as mentioned in section 3. Composition of library entities into a complete protocol stack for a group of mobile nodes can be described either textually within an initialization entity called driver, or visually using PAVE that subsequently generates the corresponding driver entity. In addition, for parallel conservative execution, it is also necessary to map the entities to processors and specify the communication topology information required by the corresponding protocol. This information may also be specified using the PAVE front-end or textually in the driver entity.

5 Parallelization

Efficient parallel simulators must address three sets of concerns:

- efficient synchronization to reduce simulation overheads;
- model decomposition or partitioning to achieve load balance;
- efficient process to processor mappings to reduce communications and other overheads in parallel execution.

5.1 Synchronization protocols

The performance of GloMoSim library is evaluated as a function of three different conservative synchronization algorithms: null message protocol [15], conditional event protocol [6], and Accelerated Null Message protocol (ANP), which is a combination of preceding two schemes [10]. The PARSEC visual front-end (PAVE) allows choice of a specific conservative runtime to be selected simply as an option in the execution command!

As the null message protocol is well known, we omit descriptions from this paper [15]. The conditional event protocol distinguishes between definite and conditional events. Definite events are those that can be scheduled locally by an entity (for instance departure of a job from a FIFO server); conditional events require communication among all the LPs or entities to identify the globally earliest conditional event, which is then converted into a definite event. Both synchronous and asynchronous algorithms have been defined to identify the earliest

conditional event; these algorithms are very similar to those used to compute GVT for optimistic synchronization protocols. The primary advantage of conditional event algorithm is that its performance is less sensitive to lookahead properties of the model [10]. Since the ANP algorithm uses null messages together with the conditional event algorithm, it is expected to perform well with models whose lookahead properties are not well understood or where they change dynamically.

A sequential simulation model must typically be refined in two ways for parallel execution: first, all the entities that comprise the model must be mapped to processors; the mapping issues are addressed in the next section. Second, to improve parallel performance with conservative protocols, the exact communication topology and lookahead properties should be specified for the model. Our GloMoSim library entities have already been added related optimization codes using appropriate PARSEC constructs. These will be briefly described in the following sub-sections.

5.1.1 **Communication topology specification.** Each PARSEC entity contains default variables called ***source-set*** and ***destination-set*** that respectively refer to a set of entities that send to or receive messages from the given entity. By specifying these sets precisely for each entity, synchronization overheads of any null message based conservative method can be reduced significantly. For this purpose, PARSEC provides following functions that can be called inside an entity to respectively add or delete entities in its source and destination sets.

add_source(entity-identifier-list)
add_dest(entity-identifier-list)
delete_source(entity-identifier-list)
delete_dest(entity-identifier-list)

PARSEC run-time system guarantees that execution of each entity will be synchronized with all the other entities belonging to its source-set. If an entity receives a message from an entity that has not been declared as its source, appropriate error messages will be generated. In our current GloMoSim implementation, the models assume a static communication topology, which is specified in an initialization entity called driver.

5.1.2 **User defined lookahead.** Seeing that good lookahead exploitation is essential to improve performance of conservative protocol [8], we try to extract lookahead from a PARSEC entity's semantic information. Specifically, if an entity's next output message has at least time-stamp increment δ compared with the entity's current simulation clock, we say a lookahead δ can be obtained. For example, in the channel propagation model, we specify the lookahead to be propagation delay because we can guarantee that a next output message will have timestamp increment of this value than the entity's current clock. Generally, if we can regard an entity simulating a network protocol as a FCFS server, lookahead can be obtained by pre-computing its service time. PARSEC provides a function called ***setlookahead(lookahead)*** to specify this dynamic lookahead.

5.2 Partitioning

The primary goal of partitioning is to decompose the simulation model into a number of components that keep the computational load approximately balanced while minimizing the communication overheads. In GloMoSim, this is attempted by assigning an approximately equal number of network nodes to each partition and an equal number of entities (or partitions) to each processor. However, in our application, complete load balance, based on decomposition of initial static network topology is impossible due to the non-uniformity of traffic directed to a partition from its neighbors. For example, assume that 2000 network nodes have been placed in an 800-m x 800-m region. The region can be partitioned in a number of ways: consider the following three partitions, respectively referred to as 4 x 4, 8 x 2 and 16 x 1 partition as shown in Figure 2, 3, 4. As each of the various partitions potentially have a different number of neighbors, the communication topology is asymmetric which implies that cross-border message traffic and hence the computational load will be unbalanced. The 2 x 8 partition or 1 x 16 partition may improve the load balance factor because the number of neighboring partitions do not differ dramatically among the various partitions as 4 x 4 partition. Experimental results of parallel performance with different partitioning schemes are presented in the next section. (Note that in these models, it is typically not appropriate to assume a toroidal region, which would smoothen out this form of communication irregularity.)

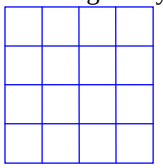

Figure 2. a 4 X 4 partition

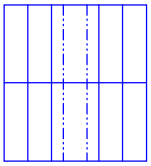

Figure 3. a 8 X 2 partition

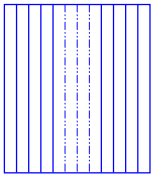

Figure 4. a 16 X 1 partition

Proper selection of partition units may reduce inter-processor messages. In our application, an inter-partition message is sent whenever a simulated network packet is transmitted with a radio power strong enough to across the border of two partitions. Otherwise, the message is just a local computation within the partition. Therefore, we select each partition to be at least equal to the effective transmission range of a mobile radio. On one hand, by increasing the partition unit size, the ratio of cross-border messages over local messages will be reduced. On the other hand, decreasing the partition size will allow nodes in distant partitions to be computed relatively independent.

5.3 Mapping

In GloMoSim, each partition consists of a set of entities, each of which simulates a certain network model executed by a group of mobile nodes. Given P partitions and M layers, this implies M x P entities. Typically this is larger than the available number of processors, so entities must be aggregated on processors using some scheme. We first evaluate the impact of ***horizontal*** vs. ***vertical*** mappings or aggregation of the entities. In the former scheme, entities from all partitions that simulate the same network layer are mapped to one processor; in the latter scheme, all entities that simulate different layers within a given partition are mapped to one processor. Considering a simple experiment that includes four different layers of protocol stacks (channel, radio, MAC, and traffic generator) under four partitions, the sequential and 4 processor parallel performance of above two mapping schemes is shown in Table 1. Although both mappings yield parallel performance to improve, the vertical aggregation is found to be more efficient for all the evaluated synchronization protocols. This follows because messages simulating data flow and control flow are passed frequently up and down among entities that simulate the corresponding layers. So, vertical mapping, which assigns entities simulating layered protocol stack for a given region on the same processor has a lower communication overhead compared with the horizontal mapping.

Mapping	GEL (sec)	4-Null (sec)	4-Cond (sec)	4-ANP (sec)
Horizontal	838	480	486	490
Vertical	839	326	381	333

Table 1 Execution time comparison of two aggregation schemes

The next concern is distribution of partitions among processors. We view the partitions as forming a two-dimensional matrix. Typical strategies [13] that have been used in decomposing matrix include random mapping, which assigns entities randomly to processors; block mapping, which assigns a block of neighboring entities to one processor (Figure 5); and cyclic mapping, which assigns entities cyclically to processors (Figure 6).

1	1	2	2
1	1	2	2
3	3	4	4
3	3	4	4

Figure 5. Block mapping

1	2	3	4
1	2	3	4
1	2	3	4
1	2	3	4

Figure 6. Cyclic mapping

Table 2 presents parallel execution time using cyclic and block mapping for 4 x 4 partition on 4 and 8 processors respectively. The network configuration is the same as used for Table 1. Block mapping achieves better performance than cyclic mapping using Null and ANP protocol, but for the conditional protocol, no significant difference can be seen.

	Processors	NULL(sec)	COND (sec)	ANP(sec)
Block	8	131	148	151
Block	4	171	154	179
Cyclic	8	147	146	160
Cyclic	4	181	152	189

Table 2: Parallel performance vs. mapping schemes

The results indicate that performance of Null message and ANP algorithms are considerably more sensitive to the specific mapping scheme (block vs. cyclic) than the conditional event algorithm. Obviously, since the model includes communication primarily among neighboring entities, block mapping is better in exploiting locality. Thus it will reduce inter-processor real message as well as piggybacked null message passing. However, conditional algorithms do not benefit from this mapping choice because its identification of the earliest event requires global communication among all entities on all processors regardless of the specific communication topology in the model. Our subsequent experiments reported in the following sections will use block mapping with vertical aggregation.

6 Experiment configurations and results

6.1 Configuration

The following configuration is used for subsequent experiments that investigate parallel performance: a raw packet traffic generator with Poisson distribution, MACA or CSMA protocol at the data link/MAC layer, a spread spectrum radio without capture ability for each node, and free space channel propagation model. The parameters in the experimental configuration are adjusted so that simulation can complete in a reasonable period. Typically, these include:

- Number of network nodes: 2000; unless specified otherwise
- Dimensions of the region over which nodes are randomly deployed: 800 m X 800 m
- Static network topology is assumed
- Traffic pattern: Poisson process (λ = 1 pkt /sec for each node unless specified otherwise)
- Radio transmission range : 50 m
- Additional parameters are defined, as appropriate, within the models for the individual entities.
- Parallel architecture: IBM 9076 SP, a distributed memory multi-computer. It consists of a set of RS/6000 workstation processors connected by a high-speed switch. Each node has a main memory of 128 megabytes.

To ensure consistent and meaningful comparisons of parallel performance, all speedup of p-processors reported in this paper are computed with respect to a P-partitioned sequential implementation (global event list algorithm implemented using splay trees) over a P-partitioned parallel algorithm on p-processors.

6.2 Partition and load balance experiment

The first set of experiments is used to identify the most suitable partitioning strategy. Figure 7 shows the variation of speedup, using up to 16 processors, for each of the three partition schemes discussed in section 5.2. Three conservative algorithms: null message (null), conditional event (cond), and combination approach (ANP) are used in each case.

We first consider the impact of the partitioning scheme: As can be seen, in general, the linear partitioning 16 x 1 yields the best performance. This is because the linear partitioning has the simplest communication topology, with each entity communicating with at most two other entities. Thus communication and null message ratio among partitions can be greatly reduced. Secondly, in all the three partitioning schemes, when running on less than 4 processors, the performance of conditional event algorithm is slightly superior to the null message algorithm. As the

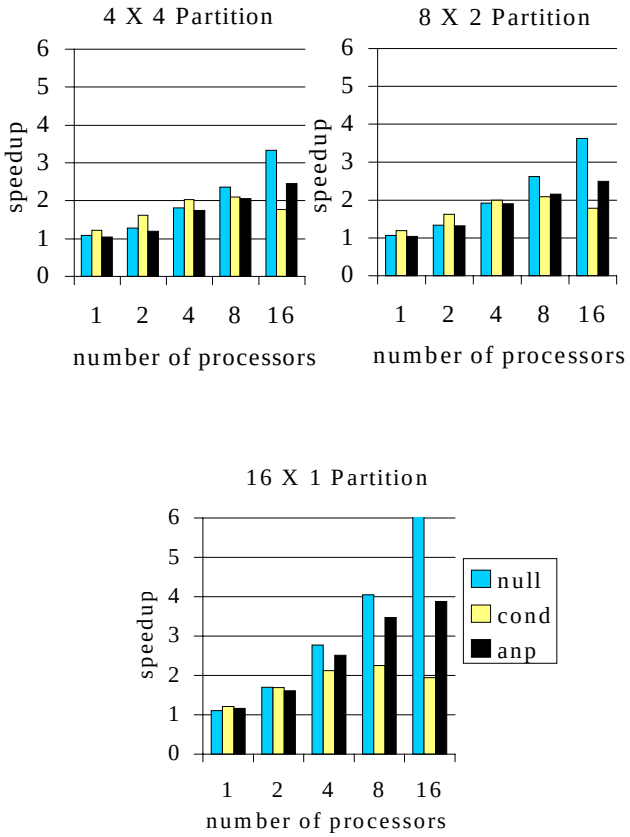

Figure 7 Speedup measurement using different partition schemes

number of processors increases, the situation reverses and the performance gap between the two algorithms gets deeper. This can be attributed to the high overhead of global communication required by the conditional event algorithm to compute the earliest event. As more processors are used, global computation used by the conditional event protocol requires more inter-processor conditional event messages to be broadcast. The performance of ANP algorithms is close to that of the null message algorithm except when the number of processors increases substantially. In this situation the overhead due to conditional event computations degrades its performance. Thirdly, note that for a given number of network nodes, for 4 x 4 and 2 x 8 partitions, the speedup does not increase significantly as the number of processors are increased beyond 4. Analysis indicates that increasing the number of processors also leads to dramatic increment of inter-processor messages, which then cancels partial potential benefits from the increased concurrency.

Partition	GEL (sec)	1-Null (sec)	1-Cond (sec)	1-ANP (sec)
4 X 4	312	286	254	308
8 X 2	315	296	265	314
16 X 1	320	276	263	288

Table 3 Execution time of sequential and one node conservative algorithm

Finally, note that sequential algorithm (GEL) is slower than 1 node conservative algorithms. Their exact execution times are given in Table 3. The relatively worse performance of GEL is primarily due to its larger entity scheduling and queue management overheads. Since GEL executes events in strictly timestamp order across all entities, this can often cause numerous (unnecessary) context switches. Whereas in the case of conservative algorithms, for processes with good lookahead, a number of events may be executed on the same entity before other events with lower timestamp are executed on a different process. This will result in fewer context switch operations. Second, GEL must sort all the entities in the order of earliest message timestamp in their input queue. Since our entity queue is implemented using a splay tree, the sorted queue operation involves more overheads than the unsorted entity lists exploited in our conservative algorithm implementation.

6.3 Parallel performance: network characteristic dependency

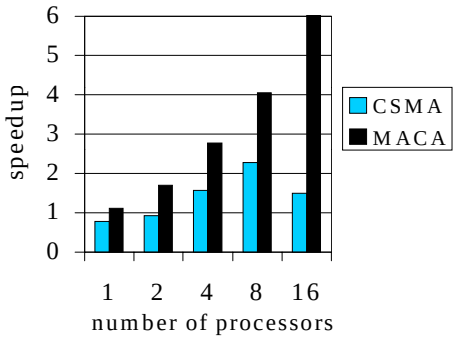

Figure 8 Speedup comparison of CSMA and MACA

As expected, different protocol models will yield different parallel performance due to their respective characteristics. Figure 8 presents parallel simulation speedup of two networks that are identical in all respects except that one of them uses the carrier sense multiple access (CSMA) whereas the other uses the multiple access collision avoidance (MACA) protocol in MAC layer. Dramatic differences are apparent in this experiment.

Analysis indicates that although both models have the similar lookahead properties, because MACA involves more real messages to negotiate in channel acquisition, the NMR ratio (null message number over real message number) is obviously smaller than that of CSMA (Table 4). As we know, smaller NMR indicates that the model has greater ratio of computation over synchronization overhead; thus potentially it has more concurrency to be exploited for parallel speedup.

Model	NMR
MACA	0.75
CSMA	3.4

Table 4. NMR comparison of two models

Considering that MACA protocol usually requires longer simulation execution times and performs better in larger wireless networks (subject to hidden terminal problems), the remaining experiments will use it as channel acquisition strategy (of course, we also like the fact that it makes our parallel performance numbers better!).

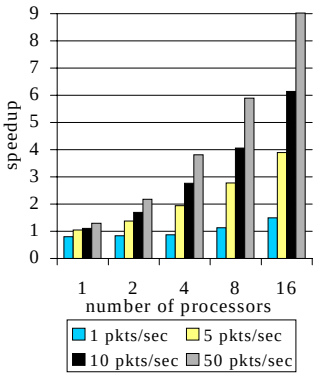

Figure 9. Speedup vs. traffic load

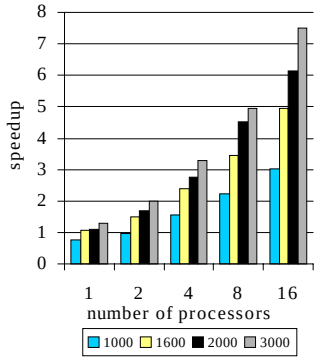

Figure 10. Speedup vs. node density

Figure 9 and Figure 10 present the impact of two network characteristics, traffic load and node density, on parallel simulation performance. Both experimental configurations are largely the same as specified in section 6.1 except that in figure 9, we vary the traffic arrival rate per network node from 1 packet/sec to 50 packets/ sec; and in figure 10, we let the node density vary by assigning different network nodes from 1000 to 3000 in a fixed geographical area. Both figures present parallel execution results on up to 16 processors using 16 x 1 partition scheme and null message synchronization protocol.

It is clear that either increasing the traffic rate or node density monotonically improves the parallel performance. The reason can be seen from the reduced NMR for both scenarios shown in table 5 and table 6.

Traffic load (pkt/sec)	NMR
1	7.7
5	1.48
10	0.75
50	0.17

Table 5 NMR of various traffic loads

Node number	NMR
1000	3.10
1600	1.24
2000	0.75
3000	0.35

Table 6 NMR of various node densities

7 Conclusion and work in progress

This paper describes ***GloMoSim***, a modular library for parallel simulation of wireless network. It has been designed to be extensible and composable: the communication protocol stack for wireless networks is divided into a set of layers, and APIs are defined for each layer. This architecture allows us to implement alternative protocols at each layer and evaluate their performance in a consistent manner. Parallel model execution is provided to users in an (almost) transparent manner. A user typically configures the experiment by visually placing icons that represent the nodes, then select sequential or one of the three available conservative synchronization algorithms. Once a parallel algorithm is selected, the analyst must additionally indicate the mapping strategy and number of processors. We have presented a set of experimental results on a distributed memory multi-computer that compare parallel performance of different conservative synchronization algorithms, partitioning schemes, entity to processor mapping and network characteristics dependency.

Current work in progress is aimed at expanding the library with new protocol modules and a thorough investigation of parallel performance under the impact of different parallel architecture environment, node mobility as well as the suitability of optimistic synchronization algorithms.

Acknowledgements

This research was supported by the DOMAINS contract funded by DARPA ITO under contract No. DAAB07-97-C-D321. Special thanks to this paper's reviewers for their valuable comments and PARSEC group for implementation support. Thanks also to Monnica Terwilliger for her assistance in preparing the camera-ready copy.

References

[1] A. Alwan, R. Bagrodia, N. Bambos, M. Gerla, L. Kleinrock, J. Short, and J. Villasenor, "Adaptive Mobile Multimedia Networks." *IEEE personal communications*, June 1997.

[2] R. Bagrodia, and W. Liao. "Maisie: A Language for Design of Efficient Discrete-Event Simulation". IEEE Transactions on Software Engineering. April 1994

[3] R. Bagrodia, Y.A.Chen, et al., "Parallel Simulation of a High-speed Wormhole Routing Network", Proceedings of PADS1996

[4] R. Bagrodia, R. Meyerr, et al., "PARSEC: A Parallel Simulation Environment for Complex System", UCLA technical report, 1997.

[5]. Christopher D. Carothers, Richard M. Fujimoto, Yi-Bing Lin and Paul England, "Distributed Simulation of Large-scale PCS Networks"

[6] K.M. Chandy and R. Sherman. "The Conditional Event Approach to Distributed Simulation." Distributed Simulation Conference, Miami, 1989.

[7] J.G. Clearly, J. J. Tsai, "Conservative Parallel Simulation of ATM Networks", Proceedings of PADS 1996

[8] R. Fujimoto. "Parallel Discrete Event Simulation." Communications of the ACM, October 1990.

[9] T. Holvoet and P. Verbaeten, "Using Agents for Simulating and Implementing Petri nets", Proceedings of PADS 1997.

[10] V. Jha and R. Bagrodia, "Transparent Implementation of Conservative Algorithms in Parallel Simulation Languages." In Winter Simulation Conference, December 1993

[11]. P. Karn, "MACA –a New Channel Access Method for Packet Radio", in ARRL/CRRL Amateur radio 9th Computer Networking Conference, ARRL, 1990.

[12] W. Liu, et al "Parallel Simulation Environment for Mobile Wireless Networks", WSC, 1996

[13] Kumar, Grama, Gupta, and Karypis, Benjamin Cumings, "Introduction to Parallel Computing"

[14] P. Martini, M. Rumekasten, J. Tolle, "Tolerant Synchronization for Distributed Simulations of Interconnected Computer Networks", Proceedings of PADS 1997

[15] J. Misra, "Distributed Discrete-Event Simulation", ACM Computing Surveys, March 1986.

[16] D. M. Nicol, "Parallel Discrete Event Simulation of FCFS Stochastic Queuing Networks." In Parallel Programming: Experience with Applications, Languages and Systems. ACM SIGPLAN, July 1988.