Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

PARTITIONING PARALLEL SIMULATION OF WIRELESS NETWORKS

Azzedine Boukerche
Alessandro Fabbri

Parallel Simulation and Distributed Systems (PARADISE) Research Lab
Department of Computer Science
University of North Texas
Denton, TX 76203, U.S.A.

ABSTRACT

In this paper, we present a simulation testbed for wireless and mobile telecommunication systems, a two-stage PCS parallel simulation testbed which makes use of a conservative scheme at Stage 1, and of time warp at Stage 2. While Time warp is considered to be an effective synchronization mechanism in parallel and distributed discrete event simulation (PDES) it is also well known for its unstability due to rollbacks and its devastating effect, i.e., serie of cascading rollbacks, among other factors. Thus, our primary goal in this paper is to study the impact and the importance of partitioning in our PCS model while reducing significantly the number of rollbacks.

1 INTRODUCTION

Parallel simulation techniques have been investigated in a number of studies to decrease the execution times of PCS (Personal Communication Service) Network simulations. Good surveys of the literature may be found in (Boukerche 1999a, Fujimoto 1990, Chandy 1979, Jefferson 1985) These techniques can be classified into two groups: *conservative* and *optimistic*. While conservative synchronization techniques rely on *blocking* to avoid violation of dependence constraints, *optimistic* methods rely on detecting synchronization errors at run-time and on recovery using a *rollback* mechanism.

Earlier parallel simulation approaches applied to PCS networks suffer from the drawback of not considering realistic models relative to time-varying mobility and call traffic, and heterogeneous geometry of the cellular layout (Carother 1994, Liljenstam 1997, Meyer 1998). Recently, in (Boukerche 99c), we have proposed a parallel simulation testbed (SWiMNet) for wireless systems that uses a hybrid approach of conservative and optimistic PDES schemes, and exploits the model independence within the PCS model. Experiments were conducted to validate the developed approach and promising results were reported. However, in the course of our experiments (Boukerche 1999c), it became apparent that many factors were affecting the execution time of the simulation, such as modeled traffic and radio resource management, cell-based partitioning, and rollback overhead.

In this paper, we present SWiMNet, a simulation framework for wireless and mobile telecommunication systems, that uses a two-stage PCS parallel simulation which makes use of a conservative scheme at Stage 1, and of time warp at Stage 2. While time warp is considered to be an effective synchronization mechanism in PDES it is also is well known for its unstability due to rollbacks and its devastating effect (serie of cascading rollbacks), among other factors. Thus, our primary goal, in this paper, is to study the impact and the importance of partitioning in our PCS model while reducing significantly the number of rollbacks. Reducing rollbacks will help to stabilize time warp. Future work will be directed at studying the other factors, such as memory management resulting from the necessity to state saving. We must point out that the performance of time warp doesn't result necessarily only in speed. Stability, reducing rollbacks (and its devastating effect), memory managements, etc.. are very important issues that any time warp developers must deal with. The primary goal of this paper is to stabilize time warp for parallel PCS models while reducing the rollback overhead in our parallel PCS simulations. In order to decrease the rollback overhead within Stage 2, we distributed statically the load of the simulation evenly. Our approach is based on an estimation of the loads involved by the model components.

The remainder of the paper is organized as follows. In Section 2, we present previous and related work. In Section 3, we describe the basic concepts of a wireless PCS network. In Section 4, we present a brief description of SWiMNet's architecture. Section 5 is devoted to a description of our partitioning scheme followed, in Section 6, by a discussion of the performance results we have obtained. Section 7 concludes the paper.

2 PREVIOUS AND RELATED WORK

Several simulation techniques have been proposed in the literature (Carothers 1994, Liljenstam 1997, Lin 1996, Panchal 1998, Zeng 1998) to speedup PCS networks simulation. *Wippet* (Panchal 1998) is a versatile simulator for wireless networks. Propagation and interference modeling at the receiver MH made simulation suitable for studying DCA schemes. Partitioning into multiple zones is done either geographical based or channel based. Channel based partitioning gives rise to better speedup due to rare synchronization of zones when a mobile changes channel. However, how that is achieved in the implementation is unclear. Selection of a channel requires interference measurement on the destination channel, which should induce overall synchronization on all zones. In (Liljenstam 1997), the authors propose another simple channel-based partitioning. In their scheme, when an MH makes a hand-off, a set of messages is sent out to all channels available on the new BS. This scheme may generate an inordinate number of messages nullifying any performance gain. Mobility of MHs are limited to constant speed and four directions – north, east, west, south. The MH is disposed of after the call is terminated. A good analysis of break-even points between cell based and channel based partitioning is reported.

The *GloMoSim* library from UCLA (Zeng 1998) supports conservative layered simulation in the context of wireless network simulation. The synchronization protocol makes use of Chandy-Misra Null-Messages scheme (Chandy 1979) Although the results reported in (Zeng 1998) show a significant reduction of the null-messages overhead, a speedup of up to 3.5 was obtained using 8 processors. Improvement on conservative simulation due to better lookahead computation is recently reported in (Meyer 1998). Low speedups shadow the improvement due to unresolved casual dependencies.

An optimistic mechanism based on time warp is proposed in (Carother 1994), and it uses uniform rectangular shaped cell logical process (LP)'s to simulate large -scale PCS networks. The mobility of an MH is limited to four neighbors only, and low blocking probability is achieved with a fixed ratio of 50 MHs/cell. Promising results were obtained with a low number of mobile hosts per cell. In (Lin 1996), another optimistic parallel simulation is presented in which the PCS coverage area is modeled by fixed hexagonal shaped cells identifying the LP. The MH's are given constant speed and direction. The obtained results were encouraging, yet the model is useful only for low call traffic and reduced mobility.

Despite the fact that promising results were obtained in these approaches, they ignored real life mobility patterns. They also restricted cell shapes to uniform geometric objects such as hexagons or squares. These limitations and weak spatial modeling of cell characteristics often simplify the simulation model, and do not capture the accuracy and realism in PCS networks performance evaluation. Linear movements have been used in most of the existing work (Carothers 1994, Lin 1996, Liljenstam 1997, Meyer 1998), whereas real movement patterns have occasional pauses and most importantly rush hour traffic and/or congested roads trigger spikes in the call arrival rate. The works reported in (Carothers 1994, Lin 1996) assumed a (fixed) ratio of MHs to channels per cell, which is unrealistic.

In our Earlier work (Boukerche 1999c), we have developed parallel discrete event modeling and simulation methodology for PCS networks. We presented a model independent simulation technique for PCS networks using a combination of *optimistic* and *conservative* paradigms. Our approach differs from previous schemes in that it exploits independence among portions of the entire PCS model, i.e., MH movements, PCS map, and call model.

In Table 1, we summarize a comparison of model-related and simulation-related issues for SWiMNet, Wippet and GloMoSim.

The problem of partitioning parallel simulation has been given a significant amount of attention. Both static and dynamic schemes have been explored for a variety of applications, such as VLSI design, traffic flow control, and queueing networks. The interested reader may wish to consult (Boukerche 1994, Boukerche 1999b) for more information.

Before we proceed further, we wish to present the basic concepts of PCS networks.

3 PCS WIRELESS NETWORKS

A mobile/wireless communication system is a wireless network that provides low-power and high-quality access to *mobile users*, or *mobile hosts* (MH). The coverage area is partitioned into *cells*, each serviced by an antenna or *base station* (BS). The cell size and shape depend on the signal strengths and on the presence of obstacles to the signal propagation. A set of BSs are controlled by a *base station controller* (BSC). The links between the BSs and MHs are wireless, whereas those between the BSs and their BSC are wireline. Several BS/BSC are connected to a *mobile switching center* (MSC), which manages all calls in a large geographical area, does call set-up and hand-off. A typical MSC is responsible for connecting as many as 100 BSs to the public switched telephone network (PSTN). The connection of PSTN and MSC can carry over 5000 calls at a time which makes it necessary to have a substantial capacity at any time.

Depending on the transmission protocol used, such as frequency or time division multiple access (FDMA or TDMA), a radio channel corresponds to a frequency or time slot. Channels can be assigned to cells according to a fixed channel assignment (FCA) scheme or a dynamic

Table 1: Comparison of Model and Simulation Issues

Model-Related Issues			
Parameters	SWiMNet	Wippet	GloMoSim
Mobility	Segmented paths	Manhattan style	Unspecified
Call arrivals	Poisson process per MH	Model-wise poisson process	Node-wise poisson process
Coverage map	Irregular cells over Voronoi diagrams	Manhattan style urban environment	Uniform geometry (hexagons or squares)
Signal propagation	Not employed	Stochastic fading	Free-space model
Call admission	FCA	RSSI based DCA	Unspecified
Handoff mechanism	Cell crossing induced	RSSI based	Cell crossing in duced
Model size	54 BSs, 10000 MHs	48 BSs	2000 nodes (short range)
Simulation-Related Issues			
Precomputation	Mobility, calls	NA	NA
Synchronization	Hybrid	Optimistic	Conservative
Partitioning	Cell Based	Zone based (channel/cell)	Static node based
Call traffic	4 calls/MH/hr	6 calls/sec to system	1 pkt/sec to each node
Speedup	11.8 on 16 processors	4 on 8 processors	6 on 16 processors

channel assignment (DCA) scheme. In an FCA scheme, each BS or cell is statically assigned a set of channels to handle calls to/from an MH residing in its signal range. On the other hand, a DCA scheme aims at improving channel utilization by reassigning channels to cells based on the traffic intensity, MH mobility, or measured interference.

When a call attempt is made to/from an MH in a cell where there are no channels available, the call is *blocked*. If an MH moves from one cell to another while a call is in progress, the MH must be connected to the destination BS (a *hand-off*). In this case, the source cell from which the MH is coming is required to release the channel that the MH is using, while the destination cell has to allocate a channel for the call to continue. If an ongoing call cannot be handed-off due to lack of an available channel, the call is *dropped*. One of the main objectives of a PCS designer is to minimize call blocking and call dropping probabilities while maximizing channel utilization.

4 THE SWiMNet FRAMEWORK

The SWiMNet is a simulation environment designed for large scale parallel simulation of wireless PCS networks. As depicted in Figure 1, the framework kernel is a graphical user interface module, which help the system designer to define a PCS model. A *Master* process within the interface module uses the defined model to configure the distributed simulator and run the simulations. Simulation results are collected by the Master and can be viewed by the PCS designer via the *Output* process.

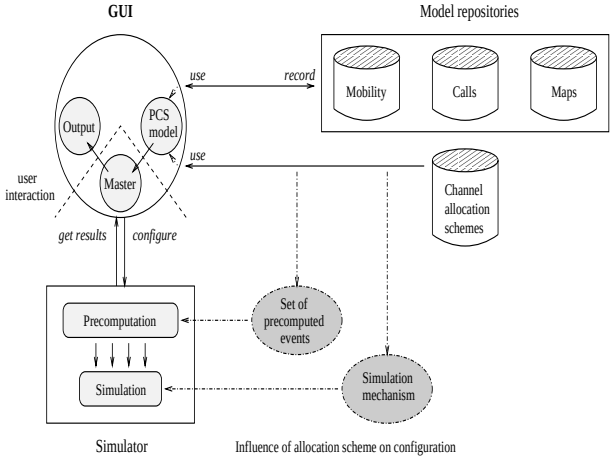

Figure 1: The SWiMNet Framework Structure

The SWiMNet involves two majors phases. In the first phase, referred to as *model definitions*, the designer defines a PCS model using several components such as (i) the mobility model; (ii) the call process; (iii) the PCS network deployment (Maps, etc..); and (iv) the channel allocation scheme to be used. SWiMNet maintains a *repository* of model components, where new components may be defined by the PCS designer as well as old components that may be reused. We choose to separate the repositories for different model components, in order to increase the flexibility of SWiMNet while defining the PCS model. Indeed, one component can be merged with several other different types, thereby creating a wide range of possible scenarios for the same component. For instance, the same PCS deployment can be tested under different traffic conditions or the same

choice of mobility and PCS deployment can be tested under different call processes or allocation schemes. However, contrary to mobility, call process and PCS deployment components, the repository of channel allocation schemes cannot be expanded by the user. Only a built-in set of schemes can be used, due to the strong influence of allocation schemes on the distributed simulator configuration. Currently, only FCA schemes are available and interference-based schemes are under study.

The second phase, called *simulation execution*, involves the actual execution of the parallel simulator on the target multiprocessor architecture. Presently, the only architecture on which the simulator has been tested is a cluster of workstations. We plan to execute it on different platforms such as SGI Origin 2000, Intel Paragon or IBM SP/2 machines.

A Visual SWiMNET component was developed using Tcl/Tk and GNUPLOT 3.5 and serves as a front-end application to facilitate the visualization and analysis of the results generated by the underlying simulator. It allows the user to readily ascertain the state of the simulator and performs detailed analysis by merely clicking a few buttons and in concord with the real-time operation of the simulator, as illustrated in Figure 2.

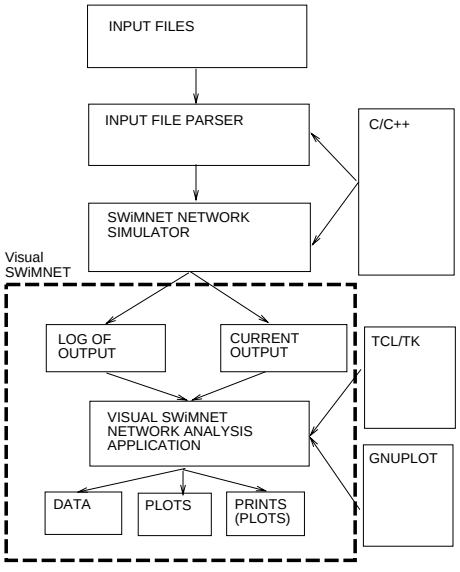

Figure 2: SWiMNET Visualisation Components

The basic idea of SWiMNet's synchronization protocol is to exploit *event precomputation*, i.e., the generation of a stream of all possible events in advance with respect to the actual simulation of channel allocations and releases. Precomputed events are then relayed to the parallel simulation part which retains only those events which can actually occur. The availability of precomputed events at simulation time reduces the causality relation within events by making causality explicit. This in turn utilizes maximum parallelism within the simulation mechanisms. Precomputation is possible in the PCS model because mobility and call related activities are mutually independent, and also the combined result is independent of the PCS model state with respect to channel management. In other words, the path of an MH through the PCS area is not influenced by the sequence of calls for the same MH, and the sequence of calls is not influenced by the MHs path. Moreover, the combination of movement and call patterns of a mobile host is not influenced by the final outcome with respect to the call allocation or call blocking/dropping. This assumption might not completely mimic reality, since in case of a blocked or dropped call a user may be tempted to try a new call immediately, contrary to what he would do if the call was successful. On the other hand, if we consider only the above possibility after a call is blocked or dropped, the final result is nearly the same since the probability that the new call attempt is instead successful, is also very low.

As shown in Figure 3, processes composing SWiMNet are grouped into two *stages*: a *precomputation stage* (Stage 1), and a *simulation stage* (Stage 2). Processes in Stage 1 precompute all possible events for all MHs assuming all channel requests are satisfied. Possible events are: (1) *call_arrival*, (2) *call_termination*, (3) *move_in*, and (4) *move_out*. Precomputed events are sent to Stage 2, where their occurrence is checked according to the state of the PCS network simulation. Stage 2 processes cancel events relative to calls which turn out to be blocked or dropped due to channel unavailability. The simulation consists in filtering the set of all the possible events, thus leaving and simulating only those which actually occur according to the channel allocation scheme chosen. A conservative synchronization protocol is used for communication between Stages 1 and 2, whereas an optimistic scheme is used within Stage 2.

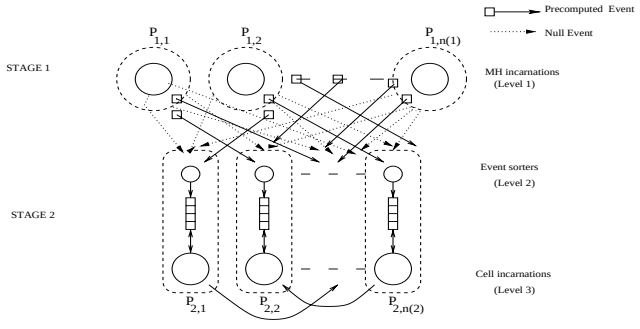

Figure 3: SWiMNet Architecture

The model partitioning in Stage 1 is mobile host-based, because each Stage 1 process produces all the precomputed events of a group of MHs. Events relative to different MHs can be precomputed independently, therefore Stage 1 processes do not interact. On the other hand, the partitioning of the model in Stage 2 is cell-based, since each Stage 2 process elaborates precomputed events occurring at a set of cells. Since events for the same call can span various cells,

Stage 2 processes need notify other Stage 2 processes of blocked calls.

Process coordination in SWiMNet is based on a *hybrid* approach, using both conservative and optimistic techniques. A conservative scheme is used for communication from Stage 1 to Stage 2: Stage 1 processes send precomputed events in occurrence time order, and periodically broadcast their local simulation time to Stage 2 processes through *null messages*. Stage 2 processes can consequently establish which precomputed events are safe to be simulated. The low percentage of blocked calls in PCS networks is exploited by an optimistic scheme for communication within Stage 2. A Stage 2 process optimistically assumes that all handed-off calls to that cell are still allocated in the previous cell. If this assumption holds, no message must be sent. Otherwise, a notification of the contrary must be sent, which may trigger a *rollback* to correct effects of a wrong assumption if the message is *straggler*.

In our mobility model, MHs are classified into groups of *workers, wanderers, travelers* or *static users*. MHs of a specific group show similar behavior. The call model is specified through intervals of different call ratios during the simulation, to represent congested hours as well as under-loaded hours. The cell topology is specified by a grid embedded in the PCS coverage area, with fine resolution of 50 meters which allows the representation of arbitrary cell sizes and shapes.

In the course of our experiments, we investigated several partitioning strategies. The primary goal of these partitioning schemes is to minimize the rollback overhead. In the next section, we describe two possible partitioning strategies for which we report results.

5 PARTITIONING SCHEMES

Our partitioning approach using SWIMNet consists of MH-based partitioning in Stage 1, and cell-based partitioning in Stage 2. The general problem can be stated as follows: given that the simulator must be composed of n_1 Stage 1 processes and of n_2 Stage 2 processes, find a partitioning of the set of n_{MH} MHs to Stage 1 processes, and of the set of n_{cell} cells to Stage 2 processes. The partition should be such that if each process is executed on a different processor, and the communication between processes is homogeneous, the global performances of the simulator are optimized.

Finding a partition of the model for Stage 1 processes which is close to optimal from the point of view of load balancing is fairly easy. In fact, MHs of a group are modeled in such a way that they have similar behavior throughout the simulation. In addition, MHs do not interact with each other. Therefore, whatever load measure is used that assigns similar loads to similar MHs, the contribution to the total load of each process given by MHs of a given group can be balanced by simply partitioning MHs of that group evenly among all Stage 1 processes. Therefore, Stage 1 is partitioned by evenly distributing MHs of the same group to all Stage 1 processes.

On the other hand, loads for different cells which are at the basis of the load for processes in Stage 2, can vary a lot. In particular, cells can be very different relatively to their coverage, the channels they are assigned, and MH movements in that area. In the following subsections we present the assignment schemes we investigated.

5.1 Round Robin Assignment

The first scheme we consider for mapping cells to processes in Stage 2 is very simple, and does not take into consideration any measure or estimation of loads. Simply, cells are assigned to the n_2 Stage 2 processes through a round robin scheme, considering cells in the order supplied by the user in the PCS model parameters. This algorithm has the advantage of being very simple, and of allowing a user to have control of how cells are mapped to processes.

5.2 Load-Based Assignment

A partition scheme of cells to processes in Stage 2 that tries to improve the round robin scheme is based on the following algorithm. The algorithm relies on an estimation of cell loads that we present further, in Section 5.3. The algorithm consists of computing an initial cell assignment, which is then improved in a sequence of steps. The purpose of this algorithm is to minimize the maximum load of all Stage 2 processes. Therefore, it tries to find an approximate solution to the NP-hard partitioning problem, where the objective function F to be minimized is:

$$F(Stage2) = \max_{P \in Stage2} Load(P)$$

where $Stage2$ is the set of n_2 Stage 2 processes, and $Load(P)$ is the estimated load of process P.

The method we chose for computing an initial solution consists of assigning cells to processes in a round robin fashion, considering cells in order of estimated load. After estimating their loads, cell numbers are pushed onto a stack. Then, cell numbers are popped from the stack and assigned to Stage 2 processes.

At each subsequent step of the algorithm, one cell is moved from the process with the largest total estimated load to the process with the smallest total estimated load. The algorithm chooses the cell to move by trying all the cells of the process with the largest load, one cell at a time. The cell movement which gives the smallest maximum load is moved at this step, and the algorithm proceeds to the next step. The algorithm terminates when no improvement is possible.

5.3 Cell Load Estimation

The load contribution of each cell to Stage 2 processes of the simulator is due to:

(i) receipt and elaboration of precomputed events; and (ii) receipt of simulation events, and consequent rollback.

If the unit of measure of the load is assumed to be the time required for receiving and elaborating one precomputed event, a fair estimation of the load for cell k is given by the number of precomputed events, E_k, plus the cost of rollbacks as number of re-elaborated precomputed events, R_k. Thus, the total load for a process P managing a set of cells $Cells(P)$ is

$$Load(P) = \sum_{k \in Cells(P)} (E_k + R_k) \ .$$

The next subsections present the estimations of E_k and R_k for a cell, given the following parameters:

- Let D be the average MH density of the cell (MH /Km^2).
- Let S be the average speed of MHs in the cell (Km/hr).
- Let L be the average call duration (hr).
- Let C be the average call traffic per MH (calls/hr/MH).
- Let T be the total simulation time.
- Let A_k be the surface area of cell k.
- Let P_k be the perimeter of cell k.

5.3.1 Estimation of Precomputed Events

Precomputed events are of four types: (1) MH move-in during a call; (2) MH move-out during a call; (3) call-arrival; and (4) call-termination.

The estimation of move-in and move-out events requires the estimation of the handoff probability at a cell. Given that the average density of MHs in the cell is known, together with their average speed and call duration, mobile hosts which contribute to handoffs are those residing close enough to the cell border that their movement brings them across the border during a call. Therefore, MHs prone to handoffs are only those inside the cell perimeter up to the distance traveled during a call. If α is the incidence angle of a MH path to the border, the MHs moving at an angle α that are prone to handoff are averagely those at a distance from the border equal to $S \times L \times \sin\alpha$. Supposing that all directions are equally likely for MHs, only half of MHs must be considered, i.e., those directed toward the border. Therefore, the average distance at which MHs are prone to handoffs is:

$$\frac{1}{2} \times \int_0^{\pi} S \times L \times \sin x \ dx = S \times L$$

Considering such a distance from the perimeter, a rough estimation of the average ratio of the MHs residing in cell k causing handoffs during a call is: $\frac{P_k \times S \times L}{A_k}$

Such a ratio must be multiplied by the ratio of time that each MH spends calling, i.e., $L \times C$. Hence, the probability H_k that a call at cell k has an handoff is given by:

$$H_k = \frac{P_k \times S \times L^2 \times C}{A_k}$$

The estimated total number of calls occurring at cell k along simulation time T, C_k, is thus $C_k = A_k \times D \times C \times T$
The total expected number of calls causing handoff at cell k is given by $H_k \times C_k$. Therefore, the total precomputed event load E_k can be estimated as

$$E_k = C_k \times (1 + H_k) \times 2$$

The multiplication by two is due to the fact that there exist two events per call (call-arrival, call-termination), and per handoff (move-in, move-out).

5.3.2 Estimation of Rollback Load

Let us denote by the number of rollbacks number of events causing a rollback. Recall that a rollback is caused by a call that was supposed to be on due to the pre-computation assumption, and turned out to be off (blocked) instead - therefore, the number of rollbacks correspond to the number blocked calls. Let us also denote by rollback load the elaboration cost per rollback. Recall that for each rollback there is a re-elaboration of those events occurring during the call that caused the rollback. It is possible that such re-elaboration involves rollbacks for other local calls, thereby further re-elaborations. However, the number of additional re-elaborations is hard to estimate analytically. Thus, we estimate the rollback load as the average number of events occurring during a call at a cell.

Since E_k is the estimated total number of events occurring during the total simulation time, the number of events occurring during a call with average call duration L, r_k, is

$$r_k = \frac{E_k \times L}{T}$$

where L/T gives the fraction of total simulation time of an average call.

Given that B_k is the blocking probability at a cell (see next subsection), the total number of blocked calls for a cell k is given by $B_k \times C_k$. Hence, the total rollback load R_k for a cell k is given by: $R_k = B_k \times C_k \times r_k$.

5.3.3 Blocking Probability

The blocking probability is computed from the Erlang-B model (Lee 1989)

$$Pr[\text{Blocking}] = \frac{Tr^{Ch}}{Ch!} \Big/ \sum_{i=0}^{Ch} \frac{Tr^{i}}{i!}$$

where: Tr is the traffic unit in Erlangs offered to the system, and is given by $\frac{Q \times L}{60}$, where Q is the number of calls per hour, estimated as $A_k \times D \times C$; and Ch is the number of channels in the cell.

6 EXPERIMENTAL RESULTS

In our experiments, we selected a real suburban city area serviced by a TDMA based PCS network. The BS deployment model was acquired from a telecommunication company. The PCS coverage map is of $11,000 \times 11,000$ meters, and is composed of a total of 54 cells. Call arrivals to each MH are a Poisson process with arrival rate C (calls/hr/MH). The average call holding time is exponentially distributed with a mean $L = 120$ seconds. A granularity of 50 meters for direction and location change is used. A number of MHs are distributed within the suburban area, with densities varying from 50 to 90 MH/Km^2 using several scenarios where 30 and 50 channels per cell were used. Concerning the mobility model, we used one group of workers, one group of wanderers, two groups of static users and three groups of travelers, located according to the presence of a town on the map. The downtown is placed at the center of the map, where base stations are more dense, and three heavy traffic highway branches are supposed to join to a loop around the town.

The goal of our experiments is to study the impact of our partitioning algorithm on the performance of SWiMNet. SWiMNet is implemented on a cluster of 16 Pentium II (200Mhz) workstations running Debian/Linux, connected via 100 Megabit Ethernet. The cluster has 8 nodes that are configurable for gigabit Ethernet connection. We made use of the LAM/MPI parallel processing system to implement message passing in the simulator. A principal motivation for using MPI is to have a flexible and easily maintainable simulator. The experimental data which follows was obtained by averaging several trail runs. We present our results below in the form of graphs of the rollback overhead as a function of the number of processors employed in the model for several arrival calls, and mobile density hosts.

As illustrated in Figures 4-5, by increasing the MH population (from 50 to 90 MH/Km^2), SWiMNet experiences in general higher rollback loads. This is evident in each figure, because curves for larger MH densities are generally above those for smaller densities. This is due to an indirect increase in the call traffic at the BSs due to the presence of more MHs. An increased call traffic results in a larger number of blocked calls, which in turn results in a larger number of precomputed events to be canceled by the simulation, hence a larger number of messages which possibly cause rollbacks.

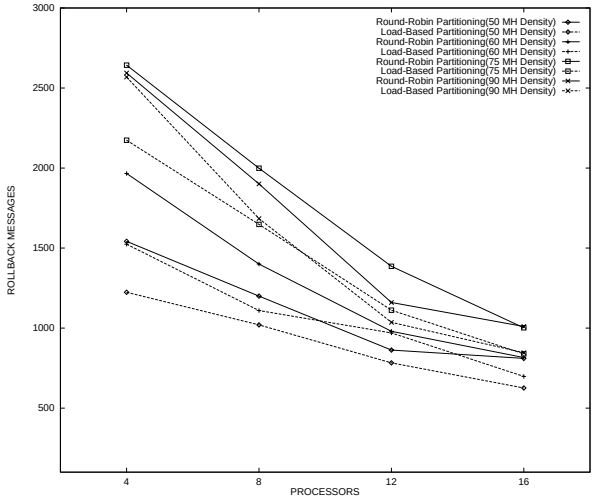

Figure 4: 30 Channels/Cell, 3 Calls/Hr/MH

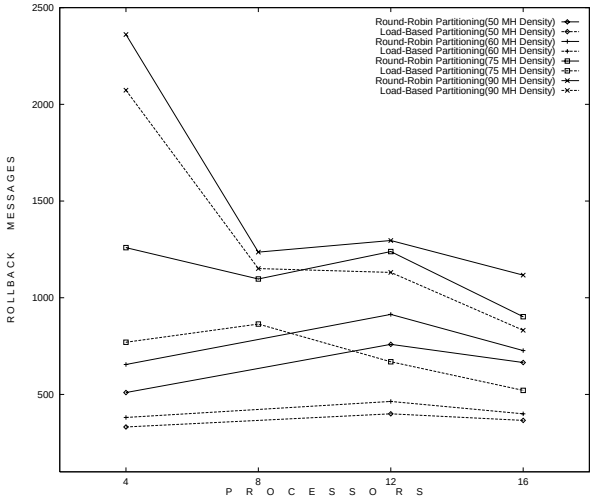

Figure 5: 50 Channels/Cell, 3 Calls/Hr/MH

Our experiments (to our surprise) indicate that the rollback messages decrease as we increase the number of processors. (This is evident in Figure 5). Since each figure represents all results obtained by using one choice of model parameters, and distributing the model over a larger number of processors. Thus, there should be more non-local messages. As consequence, the relative drift of processes and message timestamps should be larger. These causes should all increase the chances of rollbacks, on the contrary rollback loads seem to decrease. We believe that this behavior is due to the necessary rearrangement of cells to processes due to

the partitioning. We are planning to further investigate the causes of this phenomenon.

Now, let us turn to the comparison of the two partitioning algorithms. It is important to point out the improvement obtained on the rollback loads by using the partitioning algorithm based on the load estimate, over the simple round robin-based partitioning. We observe a maximum rollback load reduction around 40% with 50 channels/Cell and 50 MH density, and about 10-20 % when we use 30 channels (see Figures. 5, 4) We believe this is due to the same phenomenon we previously mentioned, i.e., the fact that ever growing additional load due to more blocked calls tends to mask the absolute improvement. One last comment concerns a qualitative observation rather than a quantitative one. By increasing the number of channels per cell (from 30 to 50), there is an increase in the number of calls/hr/MH for which the percentage improvement seems to start decreasing.

7 CONCLUSION

In this paper, we have described a parallel simulation model for wireless networks, and show that careful partitioning can have a significant impact on the efficient implementation of PCS wireless simulations. We obtained a significant improvement of the rollback overhead with the use of our partitioning algorithm (compared to the use of round-robin assignment). We plan to continue our studies on the impact of static/dynamic partitioning on PCS simulation. In particular, we shall compare our partitioning scheme with one described by Liljenstam and Ayani (Liljestam 1997). Dynamic load balancing is a critical issue for exploiting the parallelism in any applications, and particularly in parallel simulation where computational load dynamically changes with both time and space. Inter-processor communication is a basic factor to contend with on any multiprocessor machine. We would like to evaluate its relationship with the execution time of the simulation in the quest for dynamic load balancing algorithms. We also plan to investigate efficient methods to collect the data necessary for the algorithm and methods to execute the algorithm itself in parallel.

ACKNOWLEDGMENTS

The authors wish to thank the referees for their valuable comments. This work was supported by the UNT Faculty Research Grant.

REFERENCES

Boukerche, A., and C. Tropper. 1994. A Static Partitioning and Mapping Algorithm for Conservative Parallel Simulations. *Proceedings of 8th Workshop on Parallel and Distributed Simulation* 164-172.

Boukerche, A. 1999a. Time Management in Parallel Simulation. In *High Performance Cluster Computing*, Ed. R. Buyya, 375-395. Prentice-Hall Press.

A. Boukerche, S. K. Das. 1999b. Load Balancing Strategies For Parallel Simulation On Multiprocessor Machines. In *The State-of-the-art In Performance Modeling and Simulation*, Eds. G. Zobrist, J. Walrand and K. Bagchi, 135-164. Gordon and Breach Press.

Boukerche, A., S. K. Das, A. Fabbri, and O. Yildiz. 1999c. Exploiting Model Independence for Parallel PCS Network Simulation. *Proceedings of the 13th Workshop on Parallel And Distributed Simulation*, 166-173.

Carothers, C., R.M. Fujimoto, and Y. B. Lin. 1994. Distributed Simulation of Large-Scale PCS Networks. *Proceedings of MASCOTS'94*, 2-6.

Chandy, K.M., and J. Misra. 1979. Distributed Simulation: A Case Study in Design and Verification of Distributed Programs. *IEEE Transactions on Software Engineering*, 5:440-452.

Fujimoto, R. M. 1990. Parallel Discrete Event Simulation. *Communications of the ACM*, 33(10):30-53.

Jefferson, D. R. 1985. Virtual Time. *ACM Transactions on Programming Languages and Systems*, 7(3):404-425.

Lee William, C.Y. 1989. Mobile Cellular Telecommunications: Analog and Digital Systems. New York: McGraw-Hill.

Liljenstam, M., and R. Ayani. 1997. Partitioning PCS for Parallel Simulation. *Proceedings of the 5th Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, 38-43.

Lin, Y.-B., and P. Fishwick. 1996. Asynchronous Parallel Discrete Event Simulation. *IEEE Transactions on Systems and Cybernetics*, 26(4):397-412.

Meyer, R.A., and R. L. Bagrodia. 1998. Improving Lookahead in Parallel Wireless Network Simulation. *Proceedings of MASCOTS'98*, 262-267.

Panchal, J. Kelly, O. Lai, J. Mandayam, N. Ogielski, and R. Yates. 1998. WIPPET, A Virtual Testbed for Parallel Simulations of Wireless Networks", *Proceedings of the 12th Workshop on Parallel And Distributed Simulation*, 162-169.

AUTHOR BIOGRAPHIES

AZZEDINE BOUKERCHE is an Assistant Professor of Computer Sciences at the University of North Texas, and Director of the Parallel Simulations and Distributed Systems Research Laboratory (PARADISE) at UNT. His current research interests include distributed systems, parallel simulation, wireless networks and mobile computing, distributed computing, distributed interactive simulation, and VLSI designs. Dr. Boukerche has published several research papers in these areas. He was the recipient of the best research paper award at IEEE/ACM PADS'97,

and the recipient of the 3rd Collocation in the National Award for Telecommunication Software 1999 for his work on a distributed security systems on mobile phone operations, and has been nominated for the best paper award at the IEEE/ACM PADS'99. Dr. A. Boukerche was the General Co-Chair of IEEE MASCOTS'98, and ACM MSWiM'2000. He is an ACM and IEEE member, and an Associate Editor for SCS Transactions. His email address is < boukerche@cs.unt.edu > .

ALESSANDRO FABBRI received a Laurea degree in 1994 and a PhD in 1999 from the University of Bologna, Italy, both in Computer Science. He is currently a Postdoctoral Fellow at the Department of Computer Sciences at the University of North Texas in Denton, Texas. His research interests include parallel discrete event simulation, coordination languages, wireless networks, and mobile agents. His email address is < fabbri@cs.unt.edu > .