SIMULATION 64:1, 7-14
© 1995, Simulation Councils, Inc.
ISSN 0037-5497/94 $3.00 + .10
Printed in the United States of America

Modeling and Simulation of a High Speed LAN

R. Ayani, Y. Ismailov, M. Liljenstam, A. Popescu, H. Rajaei, R. Rönngren
Department of Teleinformatics
Royal Institute of Technology (KTH)
Stockholm, Sweden
Tel: +46 8 752 13 52
Fax: +46 8 751 17 93
E-mail: rassul@it.kth.se

Simulation is a tool that can be used to assess functionality and performance of communication networks and protocols. However, efficient simulation of complex communication systems is not a trivial task. In this paper, we discuss modeling and simulation of bus-based communication networks and present the results of modeling and simulation of a multigigabit/s LAN. We used parallel simulation techniques to reduce the simulation time of the LAN and implemented both an optimistic and a conservative parallel simulation scheme. Our experimental results on a shared memory multiprocessor indicate that the conservative parallel simulation scheme is superior to the optimistic one for this specific application. The parallel simulator based on the conservative scheme demonstrates a linear speedup for large networks.

Keywords: Discrete event simulation, parallel simulation, LAN

1.0 Introduction

With the growing complexity of communication networks it becomes increasingly difficult to conduct performance evaluation by strictly analytical means. It is often necessary to turn to simulation techniques to obtain adequate information on functionality and performance of communication networks and protocols.

The traditional sequential simulation techniques and tools would in many cases lead to excessive execution time (days and even weeks). Execution of such time-consuming simulation programs on parallel architectures has been an active research topic in recent years. Two main categories of parallel discrete event simulation (PDES), conservative and optimistic, schemes have been investigated by several researchers, e.g. [1, 2, 3, 4, 6].

The main purpose of this work is to investigate the applicability of parallel discrete event simulation (PDES) techniques to performance evaluation of bus-based communication networks. As a case study, we discuss modeling and simulation of a local area network (LAN) using sequential and parallel discrete event simulation methodologies. We present the results of simulating a multigigabit/s local area network, called SUPERLAN, using both a conservative and an optimistic approach. Our experimental results on a shared memory multiprocessor show that the conservative approach outperforms

1. A preliminary version of this paper has appeared in *Proceedings of the 8th Workshop on Parallel and Distributed Simulation (PADS'94)*, Edinburgh, Scotland, UK, July 1994.
2. This work was part of a research project on Advanced Simulation financed by the Swedish National Board for Industrial and Technical Development (NUTEK) under contract number 93 3319.

the optimistic one for this specific application. This work illustrates a case where the conservative scheme is preferable. The special features of the application under study (as discussed in Section 2) and the state saving policy employed in our Time Warp implementation are among the main factors contributing to the reported results.

The rest of this paper is organized as follows. In section 2 we describe the SUPERLAN and then a simulation model is developed in section 3. In section 4 we briefly describe an implementation of the optimistic and the conservative scheme. Section 5 shows the simulation results of the two approaches. In Section 6, we analyze the obtained results and finally some conclusions are presented in Section 7.

2.0 The High Speed LAN

We studied modeling and simulation of a high speed local area network, SUPERLAN, which is based on Wavelength Division multiplexing. It provides isochronous and nonisochronous services, both narrowband and broadband, with arbitrary bit rates [8]. The network has, in its first phase, a physical ring configuration with a master station (MS) and up to 64 ordinary stations (OS). Up to 16 substations are connected to each ordinary station (Figure 1). The substations are divided into three traffic classes (voice, graphics, and video), each requiring a different bandwidth for data transmission.

SUPERLAN consists of eight logically separate subnetworks, two of which are dedicated to isochronous traffic (one for data and the other for control information).

2.1 The Isochronous Subnetwork

In this study we restrict our investigations to the performance of the isochronous traffic. We simulate the Media Access Control mechanism to study the *setup delay* and *blocking probability.*

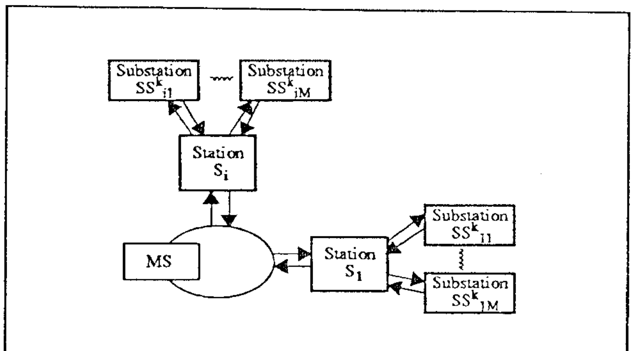


Figure 1. Structure of SUPERLAN.

The isochronous data subnetwork has a capacity of 9.6 Gbit/s. The data is transmitted in constant-length (125 µs) TDM frames with fixed-time slots (so-called universal time slots). Each slot has a length of 10 ns duration, and can provide a capacity ranging from 64 kbit/s to 768 kbit/s. Hence, the number of bits in each time slot is determined in terms of the bandwidth needed by the particular call allocated to the slot. The addresses are implicitly contained in the slot positions in the frame and determined by a call setup procedure. The isochronous control subnetwork has a capacity of 100 Mbit/s, and supports resource management flows and connection management (signaling) information flows. This subnetwork runs in a time-synchronous fashion with the data subnetwork. A train of TDM frames (also 125 µs) circulates continuously in the subnetwork, originating and ending at the master station. The first slot in each frame serves as a synchronization header. After the header follow a number of slots (chosen according to the media access mechanism) providing transportation for control units (called cells). The last slot in each frame (tail), is reserved for auxiliary functions. Every station has free access to a certain number of time slots in a frame, that is, they can read and/or overwrite control data in different time slots every time the frame passes through the node.

The master station handles network supervision and network operation control (resource allocation and management for isochronous traffic). The ordinary stations' task, on the other hand, is only to provide communication channels for the local traffic. Each of the substations connected to an ordinary station can provide only one kind of traffic.

2.2 Protocol Description

The network supports K types of traffic classes, each one having different requirements in terms of bandwidth. A static resource allocation policy is used in the master station for allocation of bandwidth resources (time slots) to different classes of traffic. Based on this protocol, each class has access to a certain number of time slots in each frame (bandwidth pool).

The multiaccess control for isochronous traffic is a mechanism acting at the call/session level. It emulates circuit switching for K traffic classes and it determines if a new requested call with a given bandwidth requirement in the isochronous data subnetwork should be accepted or denied service. This is a centralized control mechanism enforced by the master station. The call acceptance policy and the negotiated parameters are based on the traffic class (the required bandwidth), the bandwidth available for the given traffic class, and its impact on the blocking probabilities. The protocol utilizes special control units, called cells, to be transported in the control channel.

3.0 Simulation Methodologies

The main objective of the simulation study was to estimate the setup delay and blocking probability of the control subnetwork for different numbers of stations and various classes of traffic. Setup delay and blocking probability were defined as:

- Setup delay is the time taken from the instant a substation (user) generates a request for call setup to the instant when the substation receives the answer from the master station.
- Blocking probability is the ratio between the connection requests denied and the total number of requests

In this study, we investigated the following alternatives:

(a) We developed a sequential event-oriented sequential simulator and simulated the network for one second of real time. The corresponding execution time for the simulator was around two hours. Each run of the simulator (corresponding to one second real time of the network consisting of 60 stations) generated about 5000 calls and 10 million events. Clearly, one second was a short time and the obtained result was difficult to validate. A satisfactory performance evaluation of SUPERLAN would require its behavior being observed for about one hour. Considering the characteristics of the network outlined in Section 2, the sequential simulator would need weeks of execution time on a workstation.

(b) We developed two parallel simulators on our Sequent Symmetry S81 shared memory multiprocessor: one is based on a Conservative Time Windows (CTW) algorithm and the other is an optimistic Time Warp scheme. We conducted several experiments with the parallel simulators and compared the results with the sequential simulator.The results of the parallel simulators were identical to those obtained by the sequential one. Our experimental results show that the execution time of the conservative simulator (Figure 6b) was much shorter than the corresponding sequential one.

3.1 The Simulation Model

In modeling SUPERLAN, three components were essential: the master station, the ordinary stations and the substations. The communication medium was modeled as a ring circulating control information in the network. General structure of the simulation model is shown in Figure 2a.

The master station is modeled as two separate logical processes (LPs): the master station processor (MS_P) and the master station transmitter (MS_T), Figure 2b. The master station processor accepts messages sent to MS, unpacks and processes them. The transmitter generates new slots regularly, packs them into messages and sends the messages into the network.

Each of the ordinary stations is modeled as a logical process that also handles the generation of calls from the substations connected to it. To reduce the communication overhead of the simulator, the substations are integrated into the ordinary stations in the simulation model. The ordinary stations receive requests for call setup and requests for disengagement from the substations. These requests are placed in two different queues: a high priority queue for disengagement requests, and a low priority queue for call setup requests (Figure 2c). Since the substations are incorporated into the ordinary stations, messages are actually only passed between the OSs. An OS receiving a tipple may extract some information from it (e.g. response to previous request) or insert new requests to the tipple before forwarding it to the next OS.

The master station processor receives slot tipples from the last OS on the ring. Slots containing some control information (a request for connection or disconnection) are processed, and responses (to caller and called) are generated and passed to MS_T as a single cell. The transmitter queues the response information, inserts it into the next available tipple and then sends it off.

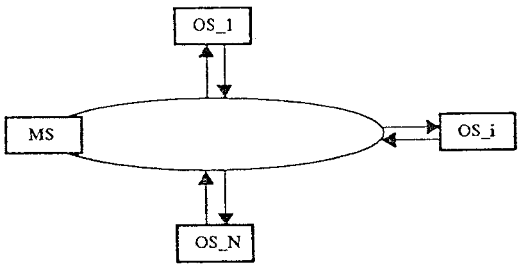


Figure 2a. Structure of the simulation model.

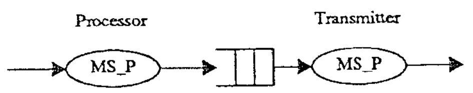


Figure 2b. The Master Station is represented by two logical processes.

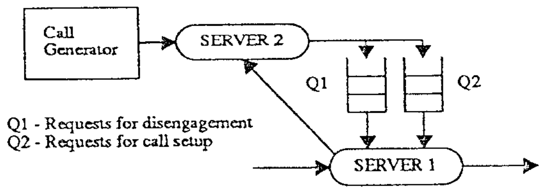


Figure 2c. Structure of an Ordinary Station

3.2 The Time Warp Implementation

The Time Warp method, proposed by Jefferson and Sowizral [8], is the most known optimistic approach. Any process can proceed as long as it has pending events, but should a process receive a message with a time stamp smaller than its local simulated time (Local Virtual Time, LVT), it has to roll back in simulated time, undoing the events executed after the receive time of the message causing the rollback. Any messages sent after this time are canceled by the sending of anti-messages. This means that the states of the processes and all "newly" received messages must be saved periodically in order to make rollbacks possible. The Global Virtual Time (GVT) is calculated as the minimum of the LVTs for all processes and the send times of all messages sent but not processed. Processes only need to keep copies of the previous states and received messages back to and including the most recent state before GVT.

The cancellation policy described above is called aggressive cancellation and is the one used in this implementation. In addition to this, Fujimoto's direct cancellation mechanism [3] for shared memory architectures is also featured. It speeds up cancellation by eliminating the need for anti-messages.

Two central ready queues for scheduling the logical processes (LPs) onto the physical processors are maintained: a high priority queue for processes that are to be rolled back, and a low priority queue for the others. The central scheduling that always selects the next pending event with the smallest timestamp should help prevent any one process from running too far ahead of the others.

```
initialization
REPEAT
    1. Nomination phase
        nominate a window Wi for each LP
        Wi=[t(ei), MAX_SIM_TIME]

    BARRIER
    2. Adjustment phase
        identify length of the windows such that
        they are independent

    BARRIER
    3. Evaluation phase
        Process the events within these windows

    BARRIER
UNTIL (cnd_of_simulation)
```

Figure 3. The conservative parallel simulation algorithm

3.3 The Conservative Parallel Simulation Scheme

Several researchers have proposed window based conservative parallel simulation schemes (e.g., see [1, 6, 9]). The main idea behind all these schemes is to identify a window for each logical process such that events within different windows are independent and thus can be processed in parallel.

We have employed the three-phase conservative time window (CTW) scheme (Figure 3), to identify independent windows. Further details on the CTW algorithm are given in [1].

The CTW scheme has the following properties:

- Events within a window are processed sequentially, but events within different windows are independent and can be processed in parallel.
- The size of each window is dynamically determined. Different windows, even those belonging to the same iteration, may have different sizes.
- Each of the phases of the algorithm may be executed by several processes in parallel. However, synchronization is required between any two consecutive phases.

4.0 Simulation Results

The following assumptions were made in running the simulators: The session time of a connection (call) is generated as an exponentially distributed random time value.

- One simulation time unit is set equal to the duration of a slot in the transmission frame, which is 680 ns.
- The distance between all ordinary stations is set equal. Thus, an integer number of slots is under transmission between two stations at any given time. The reason for this is that, to speed up the simulation, a tipple of slots rather than a single slot is passed as one message between processes in the model. This is to keep the number of messages down, as most of the simulation time would be spent on passing slots around the ring, rather than handling calls from substations since they are less frequent. The number of slots in one tipple is determined as the number of slots that can be accommodated (in terms of propagation delay) on the connection line between two stations.
- The time distance between stations was set to 1360 meters (corresponding to 10 simulation time units).

We conducted several experiments with a varying number of ordinary stations, substations, connection requests, and session times. More than 150 hours (CPU time) were spent on running these experiments. Some of the results are illustrated in Figures 4 - 9. Figures 4 and 5 show performance of the SUPERLAN (call setup delay and blocking probabilities). On the other hand, Figures 6 - 9 show performance of the parallel simulators, i.e., these figures indicate how efficient these simulators are. We will discuss some details of the simulators in Section 5.

Figure 4 illustrates some of the results of the sequential simulator. This Figure shows the call setup delay for a different number of stations and various combinations of the three traffic classes (voice, graphics and video). Since this study is focused on modeling and simulation of the network, we avoid discussing details of the SUPERLAN. Interested readers are referred to [8] for further details of the network.

Similarly, Figure 5 shows the blocking probabilities for a different number of ordinary stations and various combinations of traffic requests.

Figure 6a shows the speedup of the CTW scheme compared with one-processor execution of the same scheme. Figure 6b illustrates the speedup compared to the sequential simulator. The corresponding results for Time Warp are shown in Figures 7a and 7b. As can be seen, the conservative scheme produces almost a linear speedup compared with the sequential simulator: about

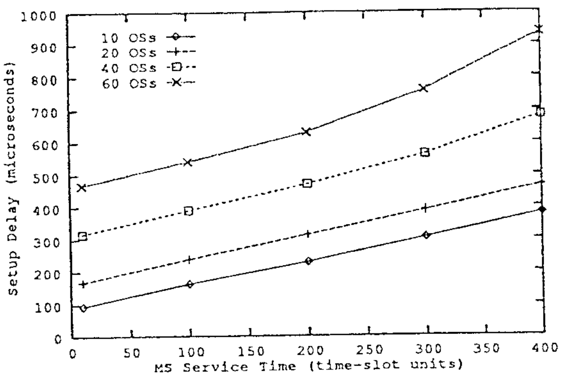


Figure 4. Call setup delay for different numbers of ordinary stations.

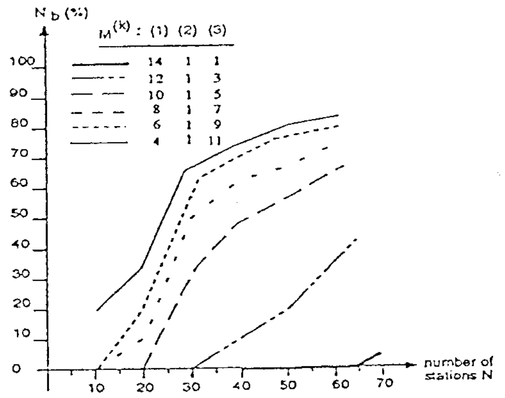


Figure 5. Simulation results showing the percentage of calls that are blocked in each time, where M(k), k=1, 2, 3 denotes the number of traffic of type k connected to each ordinary station.

6 running on 8 processors, for large networks (Figure 6b). The corresponding speedup for time warp using 8 processors is about 3 (figure 7b). In Section 5, we will discuss some of the factors affecting these performances.

Figure 8 shows the execution time of the events for various network sizes and different numbers of processors. Similarly, Figure 9 illustrates the event rate, i.e. the average number of events committed in each time unit. These two figures provide some information onefficiency of the implemented Time Warp.

5.0 Discussion

Considering the performance of the sequential simulator and the parallel ones presented in this paper, several points need further discussion. The factors affecting the reported performance figures can be classified in two categories: (a)application dependent, and (b) implementation dependent factors. Below, we will highlight the most important factors in each category.

5.1 Application Dependent Factors

The structure of the network under study, as described in Section 2, has the following properties that are worth further explanation.

- The application has a *good lookahead*[1], and this is one of the main factors behind good performance of the conservative scheme [3].
- A number of frames (each containing a fixed number of slots) circulates continuously in the control subnetwork. A certain number of frames is assigned to each substation (depending on the requested/permitted call requests. Thus, each station will access certain frames at fixed time intervals. This feature of the application is most beneficial to the conservative parallel simulation scheme (Figure 3).

5.2 Implementation Dependent Factors

As mentioned in Section 3, we implemented the parallel simulation schemes on a shared memory multiprocessor. The following factors have an impact on the performance figures.

- The major difference between the sequential and the conservative simulator on one hand and Time Warp on the other hand is in the way shared variables are treated. In the sequential and the conservative approaches shared variables do not impose any difficulties since the updating of shared variables can never be undone. In Time Warp, however, there is a

[1] *lookahead* is defined as the ability of a process to predict the occurrence of future events [3].

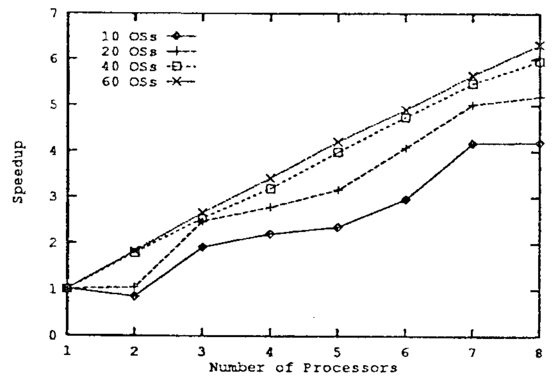


Figure 6a. Speedup of the CTW scheme compared to its one-processor execution.

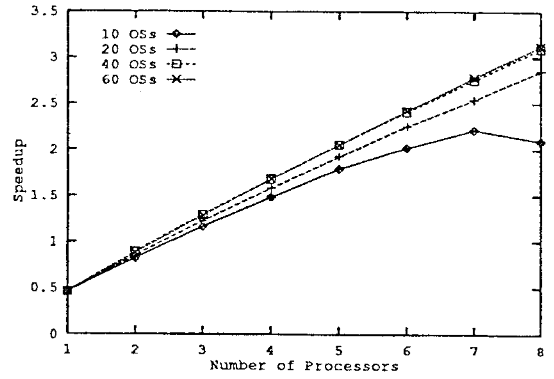


Figure 6b. Speedup of the CTW scheme compared to the sequential simulator.

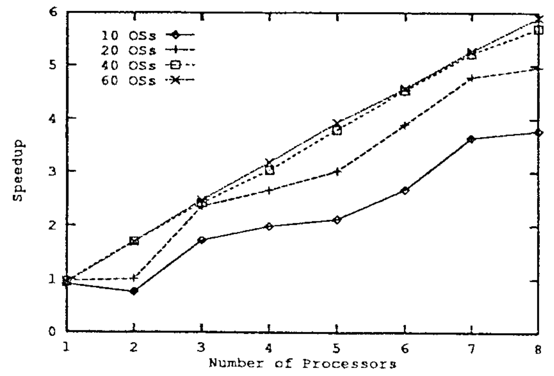


Figure 7a. Speedup of Time Warp compared to its one-processor execution.

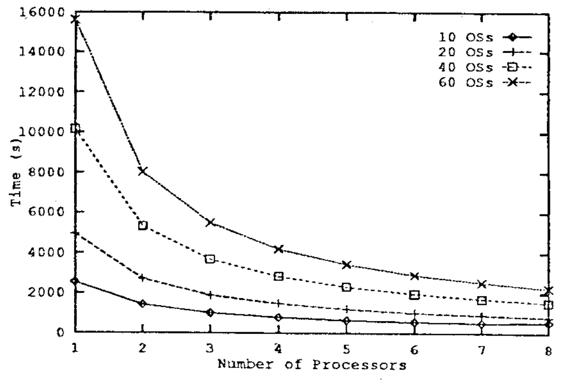


Figure 7b. Speedup of Time Warp compared to the sequential simulator.

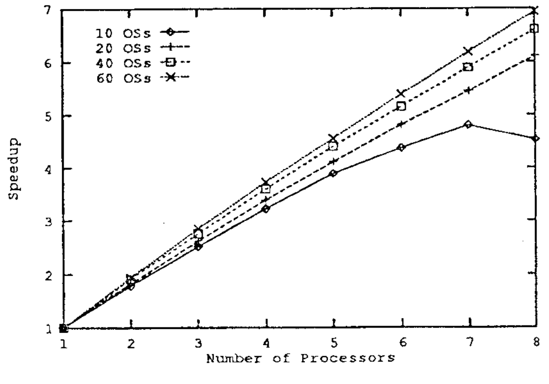


Figure 8. Event Execution time using k processors.

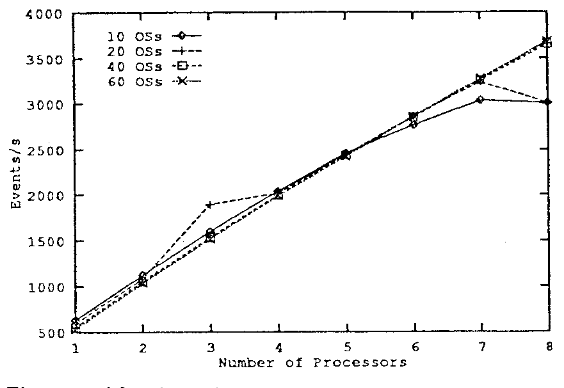


Figure 9. Number of committed events/second using Time Warp.

possibility that changes to shared variables need to be undone. In the current Time Warp implementation shared variables are not yet implemented. Thus the shared variables used in the sequential model have to be remodeled in Time Warp.

- The increased overhead in the Time Warp simulation stems mainly from the state saving and message copying. The parameters of an event message are copied before the user-defined function to process the event message is called. A pointer to the area holding the copied event is passed to this function. The copying of the parameters is needed because the user-defined function operates directly on these data and hence may change these values. In the case that a rollback occurs there must be ways of undoing such changes, i.e. by retrieving a copy of the original values. In the sequential and the conservative scheme, an event message consists of a pointer to a linked list; in Time Warp, however, the event message consists of a copy of the whole list. Consequently, an event message is about 10 bytes in the conservative and sequential cases, but about 2k bytes in the Time Warp simulation. Experiments on a 64-node fully connected network with constant message population, where the probability of forwarding a message to any other LP is uniformly distributed, indicates that increasing the state and message sizes from 32 bytes to 2k bytes results in an increase of the simulation time of the Time Warp simulator by a factor of 2 to 3.
- In this application, the state vector is rather large. Time Warp requires that the state of an LP be saved (checkpointed) occasionally. The state saving is used to restore the state of the LP if it rolls back. However, in this application rollbacks are very infrequent. This can be exploited by increasing the checkpoint interval [9, 10]. In the experiments described in this paper only every 20th state of an LP is saved. To further reduce the address space, the GVT calculation and fossil collection is performed rather frequently (four times per second). Our investigation indicates that this is almost optimal and no additional gains can be obtained by doing these operations more frequently.
- Message passing is implemented using shared memory on our multiprocessor, where the required memory for copying messages is allocated dynamically. The memory allocation is performed by the operating system using an atomic operation. Our experience indicates that memory allocation is a rather expensive operation.

6.0 Conclusions

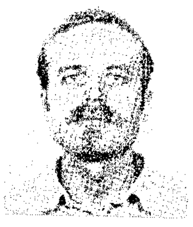
We have presented the results of simulating a multigigabit local area network using a sequential and two parallel simulators. We discussed the impact of some application and implementation dependent factors on performance of the simulators. Our experimental results on a shared memory multiprocessor show that the conservative approach outperforms the optimistic one for this specific application. Our result contradicts most of the results reported in the literature, e.g. [3]. We believe that the special features of the application, such as good lookahead, and the used simulation model were mostly beneficial to the conservative approach. We observed that it is possible to use the same simulation model and the same data structures for both the sequential and the conservative scheme and obtain a reasonable speedup. However, moving from a sequential simulator to Time Warp requires remodeling the application. Otherwise, rollback overhead and state saving costs may kill all the benefits that could be obtained using Time Warp.

References

1. Ayani, R. and Rajaei. H., "Parallel simulation based on conservative time windows: a performance study", In *Concurrency: Practice and Experience*, vol. 6(2), 119 - 142 (April 1994).
2. Fujimoto, Richard M., "Parallel discrete event simulation" *Communications of the ACM*, vol.33, no. 10, October 1990
3. Fujimoto, Richard M., "Time warp on a shared memory multiprocessor", *Intl. Conf. on Parallel Processing*, August 1989.
4. Jefferson, David R., "Virtual Time" in *ACM Trans. on Programming Languages and Systems*, vol. 7, no. 3, July 1986
5. Lin Y., et al., "Selecting the checkpoint interval in Time Warp Simulation" *Proceedings of the 7th workshop on parallel and distributed simulation (PADS'93)*, 3 - 10, July 1993
6. Lubachevsky B., "Efficient distributed event-driven simulations of multiple-loop networks" *Communications of the ACM*, 32(1):111-123, Jan. 1989.
7. Nicol D., "Performance bounds on parallel self-initiating discrete-event simulations." *ACM Transactions on Modeling and Computer Simulation*, 1(1):24-50, Jan. 1991
8. Popescu, Adrian, "A parallel approach to integrated multiGbit/s communication over multiwavelength optical networks" PhD thesis (May 1994), Dept. of Teleinformatics, Royal Institute of Technology, Stockholm, Sweden
9. Nicol D., "Performance bounds on parallel self-initiating discrete-event simulations." *ACM Transactions on Modeling and Computer Simulation*, 1(1):24-50, Jan. 1991
10. Preiss B., et al., "On the trade-off between time and space in optimistic parallel discrete event simulation", *Proceedings of the 6th workshop on parallel and distributed simulation (PADS'92)*, 33 - 42, January 1992

RASSUL AYANI received his Dipl Ing degree from the University of Technology in Vienna, Austria- (1970), Master's degree from University of Stockholm and PhD from the Royal Institute of Technology (KTH) in Stockholm, Sweden. He is an Associate Professor in the Department of Teleinformatics, Royal Institute of Technology. His current research interests are in parallel architectures, parallel algorithms and parallel simulation. He is an Associate Editor of the *ACM Transactions on Modeling and Computer Simulation (TOMACS)* and is a member of the Editorial Board of the *International Journal on Computer Simulation*. He is a member of the SCS, IEEE and ACM. His e-mail address is: rassul@it.kth.se.

YURI ISMAILOV is a researcher at the Department of Teleinformatics, Royal Institute of Technology in Stockholm. He received his Candidate of Science degree (1988) from the Department of Computer Science at the Electrical Engineering Institute in St. Petersburg, Russia. He is currently working in the area of mobile communication networks.



MICHAEL LILJENSTAM is a PhD student at the Department of Teleinformatics, Royal Institute of Technology in Stockholm. He received his MS degree in Computer Science from the Royal Institute of Technology in 1993. His research interests include parallel simulation and communication systems.

ADRIAN POPESCU received two PhD degrees in electrical engineering, one from the Polytechnical Institute of Bucharest, Romania, in 1985 and another from the Royal Institute of Technology, Stockholm, Sweden in 1994. He is an Associate Professor in the Department of Telecommunications and Mathematics, University of Karlskrona/Ronneby, Sweden. His current research interests include B-ISDN/ATM networks, communication architectures and protocols, teletraffic theory, very high speed optical networks and LANs as well as dimensioning and optimizations of future integrated networks.

HASSAN RAJAEI received his PhD in Computer Systems in 1993 from the Royal Institute of Technology, Stockholm, where he had earlier received a Licentiate of Technology. He also received an MS from the University of Utah. He is currently a member of technical staff at the Ericsson Telecom in Stockholm. His research interests include parallel simulation, distributed systems and communication systems. He is a member of SCS.

ROBERT RÖNNGREN is a PhD candidate in the Department of Teleinformatics at the Royal Institute of Technology (KTH) in Stockholm, Sweden. His research interests include sequential and parallel discrete event simulation. He received his MS degree in Engineering Physics from the Royal Institute of Technology in 1986.