
Lappeenrannan teknillinen korkeakoulu
Lappeenranta University of Technology

Jari Porras

DEVELOPING A DISTRIBUTED SIMULATION ENVIRONMENT ON A CLUSTER OF WORKSTATIONS

Tieteellisiä julkaisuja
Researcb papers 72

ISBN 978-952-214-755-4 (PDF)

Lappeenrannan teknillinen korkeakoulu
Lappeenranta University of Technology

Jari Porras

DEVELOPING A DISTRIBUTED SIMULATION ENVIRONMENT ON A CLUSTER OF WORKSTATIONS

Thesis for the degree of Doctor of Technology to be presented with due permission for public examination and criticism in the Auditorium in the Students' Union Building at Lappeenranta University of Technology, Lappeenranta, Finland on the 15th of December, 1998, at noon.

Tieteellisiä julkaisuja
Research papers
72

ISBN 951-764-270-9
ISSN 0356-8210

Lappeenrannan teknillinen korkeakoulu
Monistamo 1998

Errata

On page 1 in third paragraph second line reads: equations [*Fer94*]. ...
It should be: equations [*Nic93b*]. ...

On page 2 in fourth paragraph second line reads: *Kai90*, ...
It should be: *Cai90*, ...

On page 4 in second paragraph sixth line reads: ... Hut98b, *Lin96*, Nic93,
It should be: ... Hut98b, *Liu96*, Nic93,

On page 4 in second paragraph sixth line reads: ... SIMD machines [Aya95] ...
It should be: ... SIMD machines [*Aya93*] ...

On page 4 in third paragraph seventh line reads: Cle98, ...
It should be: *Cle96*, ...

On page 5 in third paragraph 17th line reads: [Fuj90]. ...
It should be: [*Nic88*]. ...

On page 10 in second paragraph fourth sentence reads: In the case of homogeneous ...
It should be: In the case of *heterogeneous* ...

On page 35 in third paragraph third paragraph reads: The use of tthe proposed ...
It should be The use of *the* proposed ...

On page 43 in reference [Sal93b] reads: European Simulation Symposium, ...
It should be: European Simulation *Multiconference*, ...

On page 44 in reference [Wag89a] reads: ... Simulation of Queueing: Limitations ...
It should be: ... Simulation of Queueing *Networks*: Limitations ...

The following references are missing from the bibliography:

[Lub88] Lubachevsky B.: Efficient Distributed Event-Driven Simulations of Multiple-Loop Networks, Proceedings of the 1988 ACM Sigmetrics conference, 1988, pp. 12-21.

[Lub89] Lubachevsky B.: Efficient Distributed Event-Driven Simulations of Multiple-Loop Networks, Communications of the ACM, Jan., 1989, pp. 111-124.

[Nic88] Nicol D.: Parallel Discrete-Event Simulation of FCFS Stochastic Queueing Networks, Proceedings of ACM SIGPLAN Symposium, 1988, pp. 124-137.

[Nic93b] Nicol D.: Problem Characteristics and Parallel Simulation, Parallel Computing: Paradigms and Applications, International Thomson Computer Press, pp. 418-513.

Preface

The work included into this thesis has been carried out in the Datacommunications Laboratory, Department of Computer Science, at Lappeenranta University of Technology during the years 1994-1998.

This thesis was financially supported by the Academy of Finland, Emil Aaltonen Foundation, Lahja and Lauri Hotinen Foundation, Telecom Finland and Finnish Cultural Foundation, which are gratefully acknowledged. The scholarship from the Supporting Foundation of Lappeenranta University of Technology made it possible to finish this thesis.

Several coworkers have aided me to reach this point. Matti Salmi was already doing this research when I started my own work. As Matti moved to industry Jouni Ikonen joined the research team. Without their help and the countless fruitful discussions this thesis would not be ready. Kari Heikkinen gave some valuable comments about the first versions of the manuscript. Docent Veikko Hara offered guidance on mobile networks.

My supervisor professor Jarmo Harju has always encouraged me to do this work. He took me into this research project and offered facilities for the research.

Jari Porras

Lappeenranta, November 1998

Abstract

Lappeenranta University of Technology
Research Papers 72

Jari Porras
Developing a Distributed Simulation Environment on a Cluster of Workstations
Lappeenranta, 1998

ISBN 951-764-270-9, ISSN 0356-8210 UDK 621.395.38:004.738:004.94

Keywords: Discrete-event simulation, conservative simulation, Chandy-Misra algorithm, deadlock avoidance, null messages, distributed simulation, message-passing, critical path, critical time, parallel potential, network of workstations

Simulation has traditionally been used for analyzing the behavior of complex real world problems. Even though only some features of the problems are considered, simulation time tends to become quite high even for common simulation problems. Parallel and distributed simulation is a viable technique for accelerating the simulations. The success of parallel simulation depends heavily on the combination of the simulation application, algorithm and environment. In this thesis a conservative, parallel simulation algorithm is applied to the simulation of a cellular network application in a distributed workstation environment.

This thesis presents a distributed simulation environment, Di*worse*, which is based on the use of networked workstations. The distributed environment is considered especially hard for conservative simulation algorithms due to the high cost of communication. In this thesis, however, the distributed environment is shown to be a viable alternative if the amount of communication is kept reasonable. Novel ideas of multiple message simulation and channel reduction enable efficient use of this environment for the simulation of a cellular network application.

The distribution of the simulation is based on a modification of the well known Chandy-Misra deadlock avoidance algorithm with null messages. The basic Chandy-Misra algorithm is modified by using the null message cancellation and multiple message simulation techniques. The modifications reduce the amount of null messages and the time required for their execution, thus reducing the simulation time required. The null message cancellation technique reduces the processing time of null messages as the arriving null message cancels other non-processed null messages. The multiple message simulation forms groups of messages as it simulates several messages before it releases the new created messages. If the message population in the simulation is sufficient, no additional delay is caused by this operation.

A new technique for considering the simulation application is also presented. The performance is improved by establishing a neighborhood for the simulation elements. The neighborhood concept is based on a channel reduction technique, where the properties of the application exclusively determine which connections are necessary when a certain accuracy for simulation results is required.

Distributed simulation is also analyzed in order to find out the effect of the different elements in the implemented simulation environment. This analysis is performed by using critical path analysis. Critical path analysis allows determination of a lower bound for the simulation time. In this thesis critical times are computed for sequential and parallel traces. The analysis based on sequential traces reveals the parallel properties of the application whereas the analysis based on parallel traces reveals the properties of the environment and the distribution.

List of Publications

1. Porras J., Harju J. and Ikonen J.: "Parallel Simulation of Mobile Communication Networks using a Distributed Workstation Environment", in *Proceedings of the Eurosim '95*, pp. 571-576, Vienna, Austria, September 11-15, 1995.
2. Porras J., Hara V., Harju J. and Ikonen J.: "Improving the Performance of the Chandy-Misra Parallel Simulation Algorithm in a Distributed Workstation Environment", in *Proceedings of the 1997 Summer Computer Simulation Conference* (SCSC '97), pp. 657-662, Arlington, USA, July 13-17, 1997.
3. Hara V., Harju J., Ikonen J. and Porras J.: "Simulation of Cellular Mobile Networks in a Distributed Workstation Environment", in *Annual Review of Communications*, Vol. 50, pp. 913-917, 1997.
4. Porras J., Ikonen J. and Harju J.: "Applying a Modified Chandy-Misra Algorithm to the Distributed Simulation of a Cellular Network", in *Proceedings of the 12th Workshop on Parallel and Distributed Simulation* (PADS'98), pp. 188-195, Banff, Canada, May 26-29, 1998.
5. Salmi M., Harju J. and Porras J.: "Computing the Parallel Potential of Event-Oriented Simulation Applications", in *Proceedings of the European Simulation Symposium* (ESS'94), pp. 153-157, Istanbul, Turkey, October 9-12, 1994.
6. Porras J., Ikonen J. and Harju J.: "Computing the Critical Time for a Cellular Network Simulation on a Cluster of Workstations", in *Proceedings of 12th European Simulation Multiconference* (ESM'98), pp. 61-68, Manchester, UK, June 16-19, 1998.

List of Terms

Term	Meaning
Agent	Element of Diw*orse*. Contains and executes logical processes (LPs).
Channel time	Channel time states the time for a channel inside an LP.
Connection	Direct logical channel between LPs.
Creator	Event e_1 that creates event e_2 is event e_2's creator.
Critical path	Shortest path on the event precedence graph.
Critical time	Cumulative time required on the critical path.
Event	A transaction in the simulation. LPs execute events and the execution may create new events.
Logical channel	Communication channel between logical processes.
Logical process	LP, Logical process, models a physical process for the simulation. Logical processes are executed independently and they interact by changing events or messages through logical channels.
Logical system	The physical system is modeled as a logical system for the simulation. The logical system consists of logical processes and their interactions.
Lookahead	Ability to predict future events. Necessity for the parallel and distributed simulation.
Manager	Element of Diw*orse*. Controls the start and end of the simulation. Collects information. Allows monitoring of the simulation.
Message	The simulation event is encapsulated into a message in the distributed simulation environment when it is transmitted from one workstation to another. The message is another way to express the simulation event. In the context of this thesis Message is used as a synonym for Event.
Neighborhood	Neighborhood contains the most important connections for some LP. Neighborhood is decided according to some criterion. The neighborhood concept reduces connections in the simulation model thus reducing synchronization.
Non-connected logical process	A logical process outside the neighborhood of some LP.

Parallel potential	Critical time can be used for calculating the parallel potential of the application. Parallel potential expresses the parallel properties of the simulation problem.
Physical process	PP, Physical process represents a part of the physical system.
Physical system	Problem to be simulated.
Predecessor	Event e_1 that is processed in some LP before event e_2 is event e_2's predecessor.
Process time	Expresses the simulation time of a logical process.

List of Symbols

Symbol	Meaning
T_{nm}	Timestamp of a null message
T_{om}	Timestamp of a output message
T_{ch}	Channel time
T_{p}	Process time
T_{end}	End time of the simulation
T_{cl}	Constant lookahead value used in simulation
T_{im}	Timestamp of incoming event
C_{in}	Incoming channel
C_{out}	Outgoing channel
E_{im}	Incoming event

Contents

Chapter 1

Introduction

Mobile communication networks are spreading rapidly all over the world. In Finland the density of mobile phones has exceeded the 50 % limit. The narrow band of frequencies available for these networks has forced the operators to use the bandwidth as efficiently as possible. This has led to small cell sizes and complex power control and handover procedures. The complexity of cellular systems increases all the time as new services, e.g. GPRS (General Packet Radio Service) data transmission, are developed. The 3rd generation wideband networks will increase the complexity even more.

Simulation has been used for decades for the analysis of complex real world problems. It is suitable in situations where the system is too complex for analytical approach and when the real system cannot be used for experiments [Kon91]. Therefore, simulation is a suitable method for modeling current and future cellular systems. Simulation methods can be categorized into two types:

- Continuous time
- Discrete time

In continuous simulation time evolution is typically achieved by numerically integrating time-dependent differential equations [Fer94]. In discrete-time simulation the computational effort occurs at discrete instants in simulation time, called events. Some efforts to combine these models have been made [Per98]. In this thesis discrete event simulation is applied to the simulation of a GSM (Global System for Mobile communication) mobile communication network.

The execution time of a simulation depends heavily on the accuracy and the complexity of the simulation model used. When the accuracy of the simulation model is increased or more complex problems are introduced, the model becomes more complex and more computational resources are needed. The need for resources leads in longer simulation time if no additional computing resources can be allocated for the simulation. For example, simulations of communication networks, logical circuits, etc. may require hours, days or even weeks of CPU time when sequential simulation techniques are used. For real time network or logic design this is unsatisfactory. Parallel and distributed simulation techniques seem to provide a promising approach to this problem.

The execution time of a discrete-event simulation can be decreased by using parallel computing techniques at different levels of computation. One division [Fer94] of these levels is presented in the following list:

- Application level
- Subroutine level
- Component level
- Event level

Application level execution assigns independent replications of the same simulation model with different input parameters into the available processors. This level of parallel execution is often referred to as parallel independent replications (PIRS) in the literature [Fuj90, Lin93, Won96]. The expected gain is quite high as no parallel overhead is introduced at this level. Unfortunately, this kind of a solution is not sufficient enough if the simulations are used for example for finding optimal parameters for a problem and earlier results are used as initial values for the next simulation.

At subroutine level parallel execution is achieved by distributing the subroutines of the simulation. These subroutines could include random number generation, event processing, state update, statistics collection, etc. The expected gain is limited as the number of different subroutines is quite small [Fer94]. In [Dav90, Kau87] this level is referred to as support function level.

Both component and event level execution utilize the inherent parallelism available in the physical system modeled. At the component level the emphasis is on the model components or submodels of the simulation model, whereas at the event level single events are distributed [Fer94]. Both levels consider the properties of the physical system and therefore an efficient utilization of the available parallelism is possible. In [Dav90, Kau87] these levels are called model function level. This thesis concentrates on the distributed discrete event simulation of the cellular network application at the event level.

During the past years several parallel and distributed simulation algorithms [Cha79b, Jef85, Kai90, Lub88], environments [Bag95, Cab97, Liu96], languages [Bal89, Fos95], tools [Ber85, Cub94, Dia96, Zho97], etc. have been developed in order to help the implementation process of different applications. Through proper tools the users not familiar with parallel computing concepts can reduce the run time of their simulations and gain advantage of recent development. In this thesis another distributed simulation environment is developed. Novel ideas of multiple message simulation and channel reduction enable efficient use of this environment for the simulation of a cellular network application.

1.1 Background of the work

Parallel and distributed simulation have been studied at Lappeenranta University of Technology since the year 1992. Academy of Finland financed the research project "Application of Parallel Computing to the Performance Analysis and Simulation of Mobile Communication Networks" during the years 1993-1995 and another research project, "Development of Efficient Methods for the Simulation of Large Communication Systems" during the year 1996. In the early years the research concentrated on the use of shared memory multiprocessor computers. Good results (1.5 to 1.9x speedup with 2 processors and 2.5-2.6x speedup with 4 processors) were achieved and reported [Har93b, Har93c, Sal93a, Sal93b, Har94].

At the beginning of 1995 the research concentrated on the use of networked workstations as only 2 and 4 processor computers were available. The use of a workstation cluster requires utilization of a communication network, which complicates the simulation when compared to a shared memory multiprocessor environment. The transmission delay introduced by the network slows down the simulation algorithm and can easily dominate the simulation time. The results of distributed memory workstation cluster work are presented in this thesis.

The amount and type of workstations has changed during the simulation project. In the beginning only a heterogeneous environment with various amounts of Sun IPXs, Sun Classics, Sun Sparcstations and a four processor Sun Sparcstation was available. All these machines were allocated to different people so equal load could not be guaranteed. The latest simulation experiments were performed in a group of 9 Sun IPX workstations allocated solely for the parallel and distributed simulation. This environment allowed stable loads in each workstation, which resulted in more reliable and steady simulation results.

1.2 Objectives

The objectives for this thesis were set as a change from the shared memory multiprocessor machine environment to the use of networked workstations was performed. The objectives were defined as follows:

> *To study the elements that affect the distributed simulation of a cellular network application on a cluster of workstations and to improve these elements in order to achieve as fast simulation as possible.*

A distributed workstation environment was implemented for the simulation of cellular network applications. The implemented environment allowed the study of different elements from the network components to the algorithmic modifications. Both algorithmic and structural improvements were developed in order to minimize the effect distribution. Novel ideas of multiple message simulation and channel reduction together with the known techniques like null message cancellation and LP clustering enabled efficient use of the implemented environment for the simulation of a GSM network application. This result was verified by using the proposed method of critical path analysis based on parallel event traces.

1.3 Related work

The field of parallel and distributed simulation can be divided into several distinct aspects. Each of these aspects has its own effect on the behavior of the simulation. A lot of work has been done on these research areas during the last few decades. Some of these areas were considered during the development of the distributed simulation environment. These are presented in the following list:

- Parallel architectures
- Synchronization algorithms
- Process scheduling and load balancing
- Performance prediction and analytical studies
- Applications

Synchronization algorithms

In parallel and distributed simulation a lot of work is done ensuring that events are simulated in correct order, i.e. the *causality* of events is preserved. The problem arises as the simulation of the problem parts in different processors proceed independently but the parts may affect each other. Therefore, a synchronization algorithm between these parts is needed.

Synchronization algorithms can be divided into *conservative* and *optimistic*. Conservative methods use strict synchronization as they allow simulation of only those events that are absolutely safe, i.e. can be executed without further causality errors. The Chandy-Misra deadlock avoidance algorithm with null messages [Cha79b] was the first conservative method developed. The performance of the Chandy-Misra algorithm has since been improved by several techniques [Fer94, Fuj90, Nic94]:

- Deadlock detection and recovery [Cha81, Mis86]
- Null message cancellation [Mis86, Pre91]
- Conservative time windows [Lub88, Lub89]
- Carrier null message protocol [Cai90, Woo94]
- Appointments [Nic88]

The different improvements to the Chandy-Misra deadlock avoidance algorithm with null messages have been developed because the original Chandy-Misra algorithm uses a substantial amount of synchronization messages for deadlock avoidance. In the deadlock detection and recovery algorithm frequent synchronization messages are avoided by allowing the deadlock situation and then recovering from it. This method removes unnecessary synchronization messages but it also introduces overhead due to the deadlock detection. Null message cancellation reduces the amount of null, i.e. synchronization, messages by removing obsolete nulls. This method works well if there is a sufficiently large message population, i.e. messages are not consumed at the same moment they arrive. Conservative Time Window algorithms increase some synchronization in the asynchronous simulation algorithms by introducing a time window for the processes in such a way that events within this window can be safely processed [Fer94]. One such method is Lubachevsky's Bounded Lag algorithm [Lub88, Lub89]. The carrier null message protocol increases the information carried by null messages. The goal of this method is to improve the lookahead ability though additional information and thus reduce the necessary traffic [Fer94]. Nicol has improved the lookahead ability of processes by precomputing portions of computation for future events [Fuj90]. Other studies that have improved the performance of a simulation through lookahead consideration can be found in [Lin90, Mal95, Nic96, Wag89a, Wag91]. Other approaches like conditional events [Cha89], Stimulus nulls [Dav90], Eager and Lazy blocking avoidance [Wag89b, Wag91] have been proposed for different applications. Overviews on the different conservative methods and their comparisons can be found in [Fer94, Fuj90, Nic94, Su89]

Optimistic methods allow for the simulation of any event, but when a causality error happens, events need to be rolled back and a new sequence of events is simulated. The pioneering work (Time Warp) in optimistic simulation was performed by Jefferson in [Jef85]. The Time Warp algorithm has since been improved by using different rollback and annihilation mechanisms. In lazy cancellation, processes do not immediately invalidate rolled

The following subsections present the related work done in these research areas.

Parallel architectures

Architectures used for parallel computing can be divided into several distinct classes according to various criteria, e.g. control mechanism, memory hierarchy, interconnection network or processor granularity [Kum94]. Various environments have been used for the research in the area of parallel and distributed simulation. The following subsections classify the related work into three different categories:

- Control mechanism: SIMD (Single Instruction Multiple Data) vs. MIMD (Multiple Instructions Multiple Data) computers
- Memory hierarchy: shared vs. distributed memory computers
- Availability: Multiprocessor supercomputers vs. ordinary PCs or workstations

SIMD and MIMD machines form the core of parallel computers in Flynn's taxonomy [Dun90]. Mainly MIMD type of parallel computers have been used in the parallel and distributed simulation research because of the asynchronous simulation algorithms. Especially the use of workstation clusters or networks of workstations (NOWs) [Ana97, Cab97, Car95, Du97, Mal95, Mey97, Pan97, Pea79, Sch95, Teo94, Zak97] and multiprocessor computers [Bou91, Cle96, Dri95, Fer96, Gre94, Hut98b, Lin96, Nic93, Pha98, Rön94, Zen98] has gained a lot of interest. Few experiments with synchronous simulation algorithms and SIMD machines [Aya95] can be found in the literature.

Both shared and distributed memory systems are widely used in parallel and distributed simulation. A shared memory environment allows easy and efficient implementation of a parallel simulation. Fast communication through shared memory makes the environment usable even for simulations with high communication demands. Even though the achievable speedups are good, the amount of processors available is in most cases limited to quite a small number. Examples of work in shared memory environment can be found in [Che98, Cle98, Har93a, Har93b, Har93c, Har94, Hut98b, Kel96, Kon91, Rön94, Sal93a, Sal93b, Wag89b]. The distributed memory concept makes the implementation more difficult but the amount of workstations, i.e. processors, is theoretically unlimited. In practice the usable number of workstations remains quite low as the interconnection network slows down the execution. The usability of distributed memory environments can be increased if the amount of communication through the network can be kept low. Quite recently the number of groups using networks of workstations has increased. Both homogeneous [Dia96] and heterogeneous [Cab97, Du97] systems have been suggested.

The development of multiprocessor PCs and workstations and the popularity of networked environments has increased the use of ordinary computers [Ana97, Cab97, Car95, Du97, Mal95, Mey97, Pan97, Sch95, Zak97] as opposed to the multimillion dollar supercomputers [Aya93, Dri95, Fer96, Gre94, Hut98a, Hut98b, Pha98, Nic93, Nic95, Nic96, Zen98]. This progress will eventually bring parallel and distributed simulation into every day life as working people usually have access only to ordinary computers. Governments and companies which can afford supercomputers can still benefit from the research as the results can be applied to the supercomputers.

back computations as in the original aggressive cancellation. Instead, the lazy cancellation waits to see if the reexecution of the rolled back events regenerates the same messages [Fuj90]. Optimistic time windows have the same idea of advancing computations by moving windows over the simulation time as the conservative time windows had. The goal of the time window is to limit the optimism to some extent [Fer94]. State saving plays an important role in the operation of optimistic methods. Different solutions for these problems have been proposed. State saving solutions and overviews of different optimistic methods can be found in [Fer94, Fuj90].

Process scheduling and load balancing

Problem partitioning, process scheduling and load balancing between processors play a significant role in parallel and distributed simulation, as poor partitioning and unbalanced execution of the problem reduces or even eliminates the advantage of parallel execution. Especially in distributed environments, e.g. in a network of workstations, where the communication delay is sufficiently high an imbalance in execution may have a significant effect on the results.

The goal of problem partitioning is to divide the problem into processes so that the work can be divided into the processors as equally as possible. Process scheduling makes decisions on how to distribute the processes among the processing elements in order to achieve some performance gains [Shi92]. Thus an efficient problem partition is a necessity for efficient process scheduling. Process scheduling can be performed either statically at compile time or dynamically at run time. Static scheduling is simple to perform but due to the NP-completeness of general optimal scheduling and the varying load in processing elements, only a limited success can be achieved. Dynamic scheduling improves efficiency by redistributing the processes at run time according to the information gathered from the processing elements. This redistribution is called load balancing. The goal for the load balancing strategies is to keep all the processing elements equally loaded. Several partitioning, scheduling and load balancing strategies for different applications have been presented in the literature [Ana97, Ban93, Cab97, Kon91, Nic94, Ree88, Sch95, Zak97].

Performance prediction and analysis

In parallel and distributed simulation the performance depends on several factors. Simulation algorithms, parallel architectures, applications and their implementation all have their own effect on the results. Performance prediction can be used before and after the implementation of the parallel simulator. Both methods have their own advantages.

In some cases it is necessary to know how the parallel simulation of an application would perform without actually implementing the parallel version. Several methods considering different aspects have been proposed in the literature. [Cav96] and [Dia96] present several factors that should be considered when the performance of parallel programs is predicted. [Dri95] approaches the problem through Amdahl's law. [Cub94] and [Kom96] propose tools for predicting the performance of some type of systems. All these solutions are either too general or too dedicated to certain problems to be widely used in parallel and distributed simulation. The most widely used method in performance prediction and analysis of parallel and distributed simulation systems is the critical path analysis proposed by Berry and

Jefferson in [Ber85]. It is based on an event trace which presents the events and their interactions. Critical path analysis has been used to examine the parallel properties of the application [Ber85, Tay95] and the simulation algorithms [Gun94, Jef91, Sri95].

The critical path analysis method can be used to reveal the properties of the application and the simulation algorithms. However, as sequential traces are used as a base for the analysis, some aspects are not considered at all. For example, null messages used for the synchronization in conservative algorithms have a significant role to the performance when the simulation is executed in a distributed environment with a high delay communication. Therefore, in some cases the traces need to be collected from the actual parallel simulation instead of a sequential simulation. Traces from parallel simulation can also be used for revealing the effect of architecture and process distribution to the simulation. This thesis presents a new approach to using a trace from parallel execution for the critical path computation.

Applications

Parallel and distributed simulation techniques have been used in several different applications. Queuing networks have been used widely at the development phase of simulation algorithms [Cai90, Cha79a, Cha79b, Cha81, Cha89, Dav90, DeV90, Lub88, Mis86, Nic88, Pre91] as their theory is well known. Other recently used applications contain manufacturing [Tur98], military problems [Hyb98, Tac98], communication networks [Hut98a, Hut98b, Pan98, Per98, Pha98, Zen98] and circuit simulation [Che98]. Especially the simulations of communication networks have increased lately. This is mainly due to the growing interest in the data and telecommunications field. In this thesis the distributed simulation environment is designed on the basis of the GSM network application.

1.4 Scope of the thesis

In this thesis a distributed simulation environment is implemented on top of a cluster of workstations. Several aspects affecting the distributed simulation were considered and improved while implementing the simulation environment. The following assumptions and limitations have been used:

1. The development of the distributed environment is based on the properties of the GSM network application. The structures of the environment as well as the algorithmic improvements have been designed according to this application. Other applications may not benefit from these structures and algorithms as much as the used application.

2. This thesis concentrates to the use of the conservative Chandy-Misra deadlock avoidance algorithm with null messages. Algorithmic improvements have been made by reducing communication. The excessive amount of null messages has been reduced by modifying the basic Chandy-Misra algorithm [Publication2, Publication4] and by using the null cancellation technique [Mis86, Pre91].

3. The lookahead value used by the Chandy-Misra algorithm is kept constant at 100 ms. According to the earlier research [Har93a] this value is as large as possible without a distorting effect on the application results. Even though many other research projects have

concentrated on the exploitation of maximum lookahead from the simulation applications it was felt that other aspects affecting the distributed simulation should be considered. Therefore, this thesis concentrates on the improvement of simulation environment through structural and algorithmic changes.

4. The problem partitioning used in this thesis is simple and no load balancing is implemented, i.e. static partitioning is used. In the first few experiments the effect of heterogeneous workstations was noticed in the initial phase of the simulation. Most of the subsequent simulation experiments were performed in a homogeneous workstation environment and thus adjusting was not needed. In all the experiments an ordinary 10 Mbit/s Ethernet was used as the interconnecting network.

5. Although the distributed environment has been designed according to the GSM network application, the GSM network has only been used as a simulation application on top of the implemented environment. GSM specific aspects, like channel utilization, handovers, power control algorithms, etc., have not been studied more than necessary to ensure the correct operation of the simulation environment. No GSM specific aspects have been presented in the publications, except the carrier to interference ratio in Publication 4.

Chapter 2

Distributed Workstation Environment

During the work for this thesis a distributed simulation environment Di*worse* was implemented. Di*worse* is based on the use of networked workstations. Both homogeneous and heterogeneous workstations can be used, i.e. workstations can be of the same type or of different type. Experiments have been made in both kinds of environments. The use of a homogeneous environment is preferable as no differences in computing power need to be considered. The interconnection network can be of any type. In these experiments only the results from the Ethernet environment are presented, although some experiments with the ATM network have been carried out. The following subsections present the elements, structure and operation of the environment as well as its evolution.

2.1 Logical process concept

For the purposes of parallel simulation, the physical problem needs to be divided into separate parts. In parallel simulation these parts are called *logical processes*, i.e. LPs. The conservative Chandy-Misra algorithm uses the logical system concept proposed for parallel simulation [Cha79b]. The *logical system* concept is based on the distribution of a simulation into two or more independent parts, i.e. logical processes, which can be executed on different processors. The simulation of events occurs in logical processes independently, but events may affect other logical processes, and therefore logical processes are connected via *logical channels*. All communication between logical processes is performed by interchanging timestamped events through these channels. Input and output *queues* can be attached to incoming and outgoing channels.

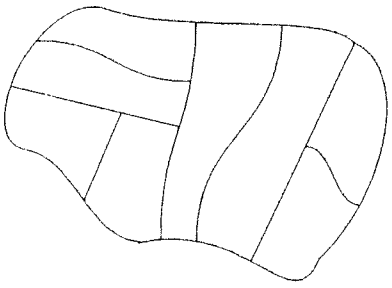

Figure 1. Partitioning of the physical problem into independent parts.

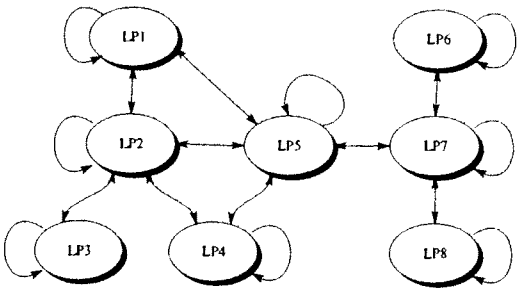

Figure 2. Logical process structure of the physical problem.

Figure 1 presents a simulation problem that is divided into 8 independent parts. The partitioning is performed according to the internal properties of the problem. Figure 2 illustrates a logical process structure for the given problem. Each independent part is modeled as a separate LP and the LPs are connected through channels. In this case the channels are created between neighboring parts of the problem. Local loop channels represent delayed actions within a given logical process.

2.2 Elements of Di*worse*

Di*worse* is based on the use of networked workstations. In this environment, all logical processes can be executed in a single workstation or they can be distributed into as many workstations as there are logical processes. In most cases there are several LPs in each workstation. In the case of homogeneous workstations, the performance of different workstations needs to be considered by the user in the initial phase of the simulation as no load balancing is implemented in the environment so far. Figure 3 presents a distribution of a simulation problem into workstations. In this distribution only four workstations are used for the environment, three for the simulation (Agents) and one for the control (Manager).

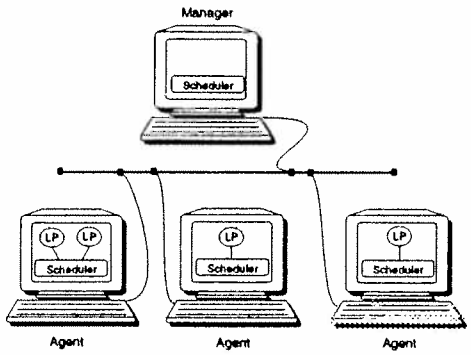

Figure 3. Division of LPs

Di*worse* is physically based on the Manager-Agent concept. *Agent* processes contain the logical processes of the simulation model, and they take care of the distributed simulation. The *Manager* process is used for controlling the start and the end of the simulation as well as error situations. It does not have any effect on the simulation, it merely synchronizes logical processes in the beginning of the simulation. All the Agent processes are executed in separate workstations whereas the Manager process can be executed in a separate workstation or in the same workstation with one of the Agents. The Manager process and the Agent processes form the base for the physical communication of the logical simulation model, since the physical communication happens between separate workstations. In the Ethernet environment, the interworkstation communication is implemented by using the TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) protocols. TCP was found to be the more suitable approach for the distribution because of its reliability.

Both the Manager and Agent processes are based on the same components. All components of the environment are implemented in such a way that they can be easily modified to fulfill different requirements. The main components of the environment are based on the logical

system concept and on the distribution concept. In the following the main components and their functions are introduced:

- Logical process
  Logical processes are the main elements of a distributed simulation. They represent both the simulation application and the distribution. The distributed simulation concept is implemented through the distributed simulation algorithm to the logical process elements. Application dependent properties can be implemented on top of this functionality. Logical processes use logical channels to transfer events in the simulation.
- Event
  Distributed simulation is based on the execution of events in different logical processes. The events are transmitted as messages from one LP to another through logical channels. The receiving LP consumes the incoming events and creates new events according to the received information. As the events need to be executed in correct order a timestamp is attached to each event. The distributed simulation algorithm takes care that the events are simulated in correct order.
- Queue
  The queue element implements the logical channels of the distributed simulation model. Different structures, e.g. a binary tree and a linked list, can be defined for different channels. Queues are used for the communication between logical processes. Incoming events are placed into the queue by the Scheduler, and they are consumed by the logical process. For outgoing messages the procedure is the opposite. Null message cancellation can be easily implemented in the queue class.
- Scheduler
  The scheduler serves as the communication element of the simulation. For each workstation participating in the simulation, a single Scheduler is needed. The scheduler sends and receives messages and directs them into appropriate queues, i.e. logical processes, according to the sender and receiver information included into the message. The scheduler also controls the execution of logical processes by using the Round Robin method, i.e. logical processes are executed by turns inside a workstation.
- Socket
  Sockets are used for implementing the communication interface between distributed elements, i.e. different physical processes. Sockets take care of the start and the end of the communication as well as the transfer of messages between the communicating processes. Socket elements for different communication environments can be implemented and thus the Socket element offers a uniform interface to the Schedulers regardless of the underlying network. In the Ethernet environment both TCP and UDP protocols have been used.

Figure 4 presents the relations between the main elements of Diw*orse*. The Manager and the Agents form a single physical process per workstation even if several LPs are run in each workstation. LPs are gathered together through the Scheduler that transmits the simulation events from the sender's output queues to the receiver's input queues. If LPs are located at the same Agent it is sufficient to move an event pointer to the receiver LP's queue. If the LPs are located at different Agents then a message containing the simulation event needs to be transmitted through the network. For this purpose the Scheduler uses Sockets.

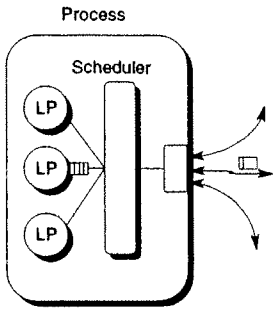

Figure 4. Relation of elements in Diworse

2.3 Layers and classes

Diw*orse* is built by using layers. The simulation application and the physical communication are separated from the main implementation of the simulation environment. This arrangement allows modification of different parts without any effect on the other parts. The layered structure of Diw*orse* is presented in Figure 5. The elements of this structure can be divided into three distinct layers, i.e. physical network, interface and implementation of the simulation services, and the simulation application.

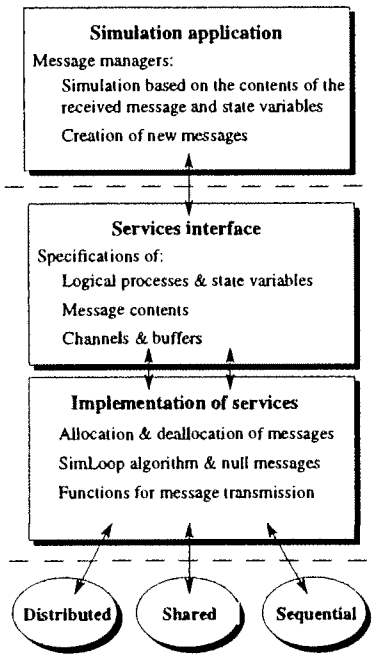

Figure 5. Layered structure of Diworse.

The simulation application level implements the logic of the simulation, i.e. it decides when to create new messages and what to do with the received messages. New applications can be easily implemented on top of the service interface by inheriting the properties of the base classes to the application specific elements. The applications use specifications given by the

interface layer and implement only the functions for the message managers. The simulation of a message executes exactly those functions which are specified for it in this layer.

The implementation of the simulation services contains functions for the distributed simulation algorithm and message transmission. The interface to the services contains specifications (base classes) for the elements, i.e. logical processes, logical channels and messages, used for the distribution. The distribution environment can be easily changed by implementing or selecting appropriate message transmission functions.

The implementation of the distributed simulation algorithm can be built to satisfy different distribution concepts, i.e. sequential, shared memory and distributed memory. The sequential approach shares the same logical process structure as the distributed and shared memory environments, but all messages are simulated by using only one logical process and its message structure. The shared memory environment differs from the distributed environment only at the message transmission functions. While the shared memory environment moves messages from one memory place to another the distributed environment sends the message over the network. The distributed environment can be easily implemented to support several network concepts, e.g. Ethernet and ATM.

Diw*orse* is based on the object oriented approach. Different elements are defined as separate classes. The main classes and their main methods are presented in Figure 6. The class structure of Diw*orse* follows the layered structure of the logical as well as the distributed system concept presented above. The communication intensive part, i.e. the Scheduler and the Socket form the base for the Manager and Agent processes. Different communication networks can be easily used by implementing different network specific Sockets. The logical process concept is implemented by using the Process, Message and Queue classes. The Message class corresponds to the events in the logical process concept. The Queue class implements the logical channels as the Scheduler delivers messages that are put into the Queue. As the logical process concept uses directed channels, both incoming and outgoing queues are needed. The operation of the Queue can be implemented by using several, e.g. list and tree, structures. The application specific part consists of the application specific processes and messages. The application specific processes contain application specific functions, e.g. power control and handover procedures in cellular network simulation. The application specific messages contain the message managers, i.e. methods that take care of the actual simulation. The message managers process the arriving message and create new messages according to the received information.

2.4 Setting up the distributed environment

When the simulation application dependent elements have been implemented and compiled the application can be run on top of Diw*orse*. The configuration of the simulation environment and the application can be set dynamically. Example 1 below presents a dynamic configuration file consisting of two Agents and a Manager.

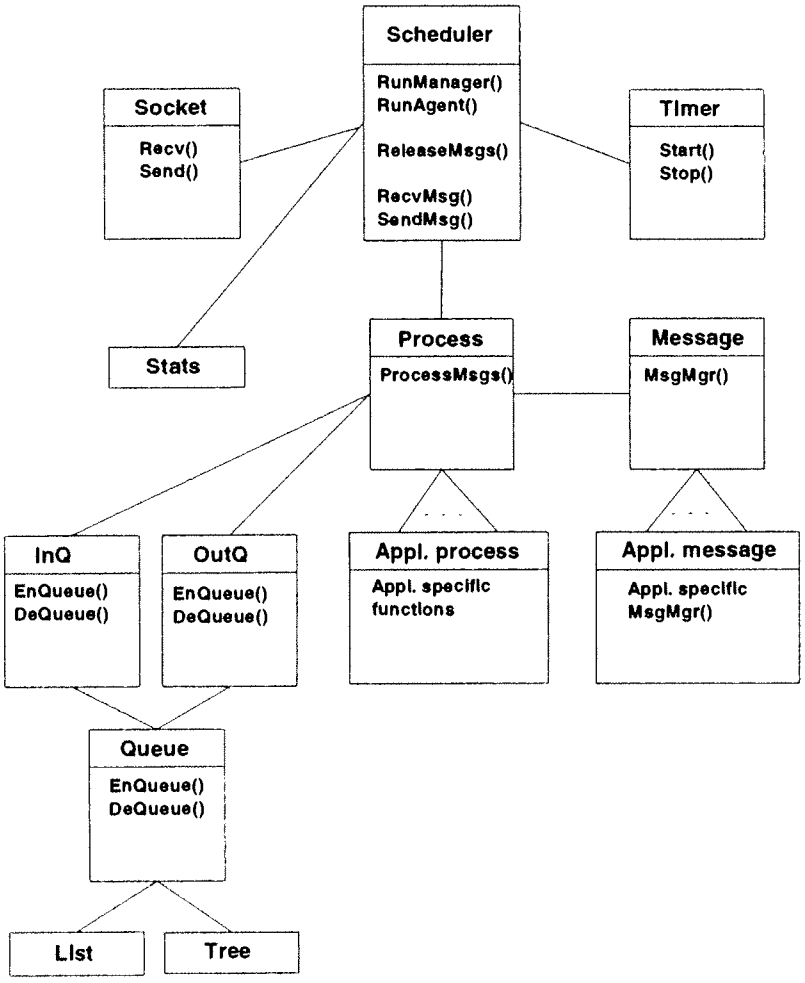

Figure 6. Element classes in Diw*orse*

```
 [GLOBAL]
// Number of sections are given here
   NumSections = 5;
// Schedulers i.e. Agents are in these workstations
   SchedLst = {
      "Sched1",      "sun1.it.lut.fi",    6031,
      "Sched2",      "sun2.it.lut.fi",    6032
   };
[Manager]
// Manager host and port
   MHost = "sun1.it.lut.fi";
   MPort = 6030;
[LP1]
   Sched = 1;
   NeighLst = {2,3};
 [LP2]
   Sched = 2;
   NeighLst = {1};

[LP3]
   Sched = 2;
   NeighLst = {1,4};
 [LP4]
   Sched = 1;
   NeighLst = {3};
```

Example 1. **Configuration file for setting the physical locations of the LPs.**

Different aspects of the simulation environment, e.g. the physical locations of simulation processes, can be set dynamically. The use of dynamic configuring allows changes in configuration files without the compilation of the code. In Example 1 the configuration file defines logical process locations, i.e. physical distribution, and connections. In this case the configuration file defines two Agents and a Manager. The Manager and one of the Agents are executed at the same workstation (`sun1.it.lut.fi`). The distributed simulation consists of four LPs. LPs 1 and 4 are located in Scheduler1 whereas LPs 2 and 3 are located in Scheduler2. Other configuration files can be used for setting up the parameters for the simulation application. For example in cellular network simulation, these files include information about locations of base stations, channels used for the communication, transmission powers, etc.

2.5 Evolution of the environment

The development project started in the beginning of 1995. At that time a shared memory version of the cellular network simulator was implemented but the lack of suitable multiprocessor environments restricted the usability of this simulator. Therefore, a change from a shared memory multiprocessor environment to a distributed workstation environment

was carried out. Di*worse* is a result of this evolutionary project. Different aspects of the distributed environment were studied one at a time and the developed improvements were implemented into Di*worse*. The work can be divided into four parts. These parts and their contents are presented in the following:

Phase 1: Basic operations - year 1995

- Several workstations
- UDP communication
- Each LP a separate process
- Basic Chandy-Misra algorithm

In the first phase the general idea of a distributed workstation environment was studied. The use of several workstations was based on the UDP protocol and each LP was modeled as its own physical process. This first approach was based on the same basic Chandy-Misra parallel simulation algorithm that was used in the shared memory version. The cellular network application was transferred from the shared memory environment to the distributed environment. Few changes were required as the use of the shared memory environment allowed pointer transfers. The work consisted mainly of the implementation of known techniques. The results are presented in Publication 1.

Phase 2: Improvement of communication through algorithmic changes - year 1996

- Improvement of the Chandy-Misra algorithm
- Improvement of the communication by using multicasting

In the second phase the excessive load generated by the basic Chandy-Misra was considered. Through the improvement of the simulation algorithm the amount of messages was reduced and the simulation was accelerated. The improved Chandy-Misra algorithm, SimL∞p, was achieved by allowing simulation of several messages in each simulation loop. The algorithmic improvements were partly based on known techniques, e.g. null message cancellation, and partly on new ideas, i.e. multiple message simulation. The use of UDP communication allowed improvement of communication through the multicasting technique. Through multicasting the amount of messages in the Ethernet network was reduced. Publication 2 presents the results achieved in this phase.

Phase 3: Improvement of communication through structural changes - year 1997

- Implementation of the Scheduler element
- Combination of LPs in one workstation into a single physical process
- Communication by using TCP

As the communication was improved in the second phase through algorithmic changes, the structure of the simulation environment became the bottleneck of the simulation. Therefore, several LPs were combined into one physical process through the Scheduler element. The use of LP clustering improved the simulation, as single workstations did not need to consider different physical processes. At the same time the communication method was changed from the UDP protocol into TCP protocol. Publication 3 presents some of the work done in this phase.

Phase 4: Consideration of the application - years 1997-1998

- Reduction of logical channels in the SimL∞p algorithm
- Analytical studies

In phase four the research concentrated on the application specific parts of the simulation environment. The idea of simulation acceleration through channel reduction was developed as the simulation model for the GSM network was studied. Although channel reduction can be applied to any application, the method is highly application dependent as channel reduction reduces the accuracy of the simulation model. The channel reduction concept presents a novel approach to speed up parallel simulation. The results of this research are presented in Publication 4.

The distributed simulation problem was also approached by using analytical methods. The analytical methods were used to study the parallel properties of the simulation application. Publication 5 is based on the work done in the shared memory environment and reveals the properties of the simulation application. Publication 6 extends the analytical examination into the distributed workstation environment. In this paper a novel idea for analyzing the effect of distribution in distributed simulation is presented.

2.6 Discussion

Networks of workstations (NOWs) over a local area network are increasingly being used as a distributed simulation environment. This type of environment offers a cost-effective platform for parallel computing. As the local area networks have not been developed for parallel processing the communication costs remain quite high. This places severe constraints on obtaining performance gains from this environment [Pan97]. This thesis presents different aspects that need to be considered when implementing a distributed simulation environment on a network of workstations.

The main goal in the use of a distributed environment is to get some performance gains, e.g. acceleration of the simulation, over the sequential approach. The high communication costs of the distributed environment tend to make this objective hard to reach. Therefore, careful consideration should be given to the problem of distribution. The effect of the communication network on performance can be minimized by providing faster interconnection networks or by reducing the amount of communication. The first approach contains the use of new networking solutions, e.g. ATM, FDDI and fast Ethernet, as well as provision of custom interconnects, e.g. Myrinet [Pan97]. In this thesis the approach for reducing the slowing effect of communication is based on the second method, namely the reduction of required communication. This reduction has been achieved with structural and algorithmic changes.

Structural changes like LP clustering have been used for improving the efficiency of the simulation [Bou91, Mal95, Sim95]. LP clustering reduces the communication through the network by moving some of the communication to happen within a cluster of LPs, i.e. within a single workstation. Unfortunately, it also reduces the achievable parallelism as LPs within a cluster cannot be executed in parallel. However, LP clustering was successfully used in this

thesis. Another structural modification could be the merging of several messages into a single message. This method reduces the communication overhead per message as the constant communication cost is higher than the additional cost of sending a longer message. For small messages the merging would not necessarily introduce any additional overheads. This method works well if the message population is sufficiently high. With low message population the message merging could introduce additional waiting delay when some LPs wait for the next event. Message merging is still to be tried out in our distributed simulation environment. The multicasting technique works like message merging but multicasting allows the sending of the same message to multiple receivers with only one physical message. Moderate results were achieved with this technique.

Algorithmic changes are another way of affecting the communication. The amount of messages required for the simulation were reduced by using null message cancellation and multiple message simulation (see results from Publication 2). Both of these methods affect the amount of synchronization, i.e. null messages. A method considering the application, i.e. channel reduction, was also developed. Channel reduction is based on a novel idea of LP neighborhoods. This method reduces LP synchronization thus allowing faster simulation. Algorithmic improvements are considered more thoroughly in Chapter 3 below.

The efficiency of the distributed simulation environment can also be improved by other means than reducing communication. Process scheduling and load balancing are important aspects that should be considered. Good results cannot be achieved if the work is unevenly divided among the workstations. In this thesis a static scheduling of LPs has been used. It was felt that the work can be divided equally enough. However, both the analytical approach and the real simulations (Publication 6) show that in some cases load balancing could improve the execution.

Chapter 3

Development of the Simulation Algorithm

In parallel simulation the simulation algorithm plays an important role as the LP synchronization and the possible deadlock situations need to be solved. Synchronization of distinct LPs is required in order to preserve the correct simulation. Deadlock situations may happen if LPs, somehow, start waiting for each other. In this thesis these aspects are included into the basic Chandy-Misra simulation algorithm used in the first implementation of the simulation environment. In the Chandy-Misra deadlock avoidance algorithm deadlock situations are totally avoided by using special synchronization, i.e. null messages. In this thesis further development of the simulation algorithm has reduced the amount of synchronization while keeping the algorithm deadlock free. The initial work, i.e. implementation of the Chandy-Misra algorithm, and the null message cancellation are based on work done by others. Other modifications, i.e. multiple message simulation in SimL∞p and the neighborhood restriction, have been developed and published by the author of this thesis. The following subsections present the work done in the development process of the simulation algorithm.

3.1 Basic Chandy-Misra deadlock avoidance algorithm

The basic Chandy-Misra algorithm uses the logical process structure of the parallel simulation model. Logical processes are connected by logical channels and they communicate by interchanging timestamped messages through these channels. The logical processes maintain a local *process time* T_p for stating the local time of the simulation. They also maintain a *channel time* T_{ch} for each incoming channel. Channel time equals the smallest timestamp of the messages in the incoming queue attached to the channel or, if the queue is empty, the timestamp of the message last read from the channel. A logical process repeatedly searches for the incoming channel with the smallest channel time. If the selected channel has a message, it is read and processed, otherwise the logical process continues searching for channels until a message is available. The Chandy-Misra algorithm is presented in Algorithm 1.

The Chandy-Misra algorithm ensures that messages are simulated in correct order and no deadlocks occur in the simulation. Chandy and Misra have shown in [Cha79b] that deadlocks can be avoided by sending a special null message to each output channel after the simulation of a real message. The null message contains only timing information, i.e. *timestamp*. The timestamp of a message reveals how far the simulation has proceeded, i.e. reception of a null message with a timestamp T_{nm} ensures that the logical process which sent the null message, certainly does not later send any messages with timestamps $T_{om} < T_{nm}$. The simulation of a

received null message does not change the state of the logical process or cause any other operations than new null message transmissions.

Algorithm 1: Basic Chandy-Misra algorithm

```
1     T_p = 0
2     WHILE T_p < T_end DO
3       FORALL incoming channels C_in
4         Enqueue NewMsgs(InQ)
5         Update channel time T_ch
6       ENDFOR
7       Select the incoming queue InQ with smallest channel time T_ch
8       IF IsMsg(InQ) THEN
9         E_im = Dequeue NextMsg(InQ)
10        T_p = T_im
11        Simulate(E_im)
12        Enqueue NewMsgs(OutQ)
13        FORALL outgoing channels C_out
14          ReleaseMsgsUpTo(T_p+T_cl)
15          SendNullMsg(T_p+T_cl)
16        ENDFOR
17      ENDIF
18    ENDWHILE
```

In a conservative simulation a logical process i cannot send a message to another logical process j until it can be sure that no messages with a smaller timestamp will be sent in the future. As a result of this some amount of simulated time need to elapse before the message can be safely transmitted. This amount of time depends on the ability of the process to "look ahead" into the future [Fuj89]. Chandy and Misra have stated in [Cha79b] that in some cases it is possible to deduce the histories of messages that will be sent from logical process i to some other logical process j up to some time t' solely from the histories of messages received by logical process i up to some earlier time t. In this case messages sent by logical process i to logical process j in the interval (t,t') are independent of messages received by logical process i after t. Thus lookahead ability can be considered as a property of the simulation model to predict the future events. In this thesis lookahead is implemented by using the constant timestamp increment t_{cl}.

3.2 Null message cancellation

Null messages are a necessity for the operation of the Chandy-Misra algorithm as they synchronize the LPs and take care that deadlocks cannot occur. However, these synchronization messages may have a significant effect on the performance of the Chandy-Misra algorithm. Especially in the distributed environment an excessive number of null messages slows down the simulation as the messages need to be transmitted through a relatively slow transmission network. Null message cancellation [Mis86, Pre91] is one of

the existing techniques to reduce the effect of null messages. In this thesis null message cancellation is used with our own improvements, i.e. multiple message simulation and channel reduction.

Null message cancellation efficiently reduces the amount of null messages in the Chandy-Misra algorithm (see Publication 2). This method is based on the fact that null messages contain no other information than a timestamp. Therefore, several subsequent non-simulated null messages can be reduced into a single null message containing the timestamp of the last null message. Null message cancellation can be implemented in several ways. The messages themselves can be canceled, in which case there must be functions which remove obsolete null messages from the queue structure whenever necessary. Another way to implement the null cancellation is to have a single variable telling the latest timestamp of all null messages for each channel. This method does not need to handle queue structures and thus can be implemented easily. In this thesis null message cancellation is achieved by using the variable approach.

3.3 SimL∞p algorithm

The performance of the Chandy-Misra algorithm can be improved through existing techniques like null message cancellation. However, null message cancellation affects only the null messages but not the real messages. In this thesis the simulation of real messages is accelerated by using the multiple message simulation technique developed by the author.

In the basic Chandy-Misra algorithm the main loop (Algorithm 1: lines 2-18) is traversed for each message to be simulated. Therefore, the message receive (Algorithm 1: lines 3-6) and release functions (Algorithm 1: lines 13-16) are performed in each simulation loop. These functions generate a moderate overhead as only minor changes are expected to happen during the simulation of a single message. The overhead caused by these functions can be decreased by allowing the simulation of several messages within a single simulation loop. This multiple message simulation technique reduces unnecessary channel updates but delays the release of new messages. Both the null message cancellation and the proposed multiple message simulation techniques were implemented to our distributed simulation environment Diw*orse*. These two modifications form an efficient simulation algorithm SimL∞p. The SimL∞p algorithm is presented in Algorithm 2.

The main idea in the SimL∞p algorithm is that every logical process simulates messages as long as there are messages available. In each simulation loop new messages are received and channel times are updated (Algorithm 2: lines 3-6), available messages are simulated (Algorithm 2: lines 7-16) and the created messages are released (Algorithm 2: lines 17-22). In the proposed algorithm the multiple message simulation is achieved by an internal loop within the main loop. The use of an extra channel update (Algorithm 2: line 14) within this internal loop makes it possible to simulate several messages without a message reception or transmission. Due to this structure the logical processes can simulate events up to the current process time plus lookahead before new messages need to be released.

Algorithm 2: SimL∞p algorithm

```
1     T_p = 0
2     WHILE T_p < T_end DO
3       FORALL incoming channels C_in
4         Enqueue NewMsgs(InQ)
5         Update channel time T_ch
6       ENDFOR
7       DO
8         Select the incoming queue InQ with smallest channel time T_ch
9         IF IsMsg(InQ)
10            E_im = Dequeue NextMsg(InQ)
11            T_p = T_im
12            Simulate(E_im)
13            Enqueue NewMsgs(OutQ)
14            Update channel time T_ch of the InQ
15        ENDIF
16      WHILE MsgAvailable
17      FORALL outgoing channels C_out
18        ReleaseMsgsUpTo(T_p+T_cl)
19        IF no message released to C_out
20          SendNullMsg(C_out,T_p+T_cl)
21        ENDIF
22      ENDFOR
23    ENDWHILE
```

The simulation process advances by executing the simulation loop, where the logical process traverses through channels according to the channel time and simulates messages available in the selected channels. In the beginning of the loop the logical process has a process time T_p and the last message sent by the logical process has a timestamp of current process time plus lookahead, i.e. $T_p + T_{cl}$. This message is sent in the end of the previous main loop of the simulation. If the lookahead is the same for all logical processes then the logical process may receive messages from other logical processes with timestamps up to the current process time plus lookahead plus lookahead, i.e. $T_p + T_{cl} + T_{cl}$. This is two lookaheads greater than the current process time which means that messages with higher timestamps than the current process time plus lookahead need to be simulated in the next simulation loop. The restricting element here is the queue to the logical process itself. The latest timestamp received by the logical process from itself is the current process time plus lookahead, i.e. $T_p + T_{cl}$. Therefore, the logical process cannot proceed beyond this time and it must release outstanding messages. These restrictions are illustrated in Figure 7. In optimal conditions, several messages can be processed within a single simulation loop, which may last as long as the time of one lookahead.

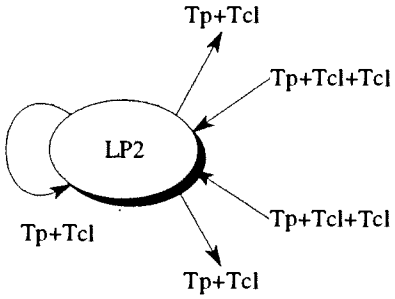

Figure 7. Restriction of execution in SimL∞p.

The SimL∞p algorithm allows for the simulation of several messages before any created messages are released. If this method is compared to the original method where the messages are released after each simulated message the following aspects should be considered:

- synchronization overhead, i.e. number of null messages is reduced
- real messages may be unnecessarily delayed

These aspects have opposite effects to the performance of the simulation. The reduction of null messages improves the distributed simulation as a smaller amount of messages need to be transmitted between workstations. Parallel overhead is also reduced as certain functions, e.g. channel update, need to be executed less frequently. The negative effect of the SimL∞p algorithm is the additional delay introduced to the created real messages. This delay is generated during the main simulation loop as the created messages need to wait until all possible messages are simulated. The introduced delay is meaningless if the population of simulation messages is sufficient, i.e. LPs have enough work to do while these new messages are kept at the creator LP. For low message population the delay may affect the simulation as LPs are waiting for the next message to arrive.

3.4 Neighborhood concept and the reduction of logical channels

Both the basic Chandy-Misra and the SimL∞p algorithm are based on the use of a logical process model. According to the model, an LP has logical channels or *connections* to all those LPs that need to communicate with it. The Chandy-Misra parallel simulation algorithm uses these logical channels for the synchronization of the connected LPs. The synchronization in the Chandy-Misra algorithm restricts the overall progress of the simulation as connected LPs must proceed within a certain time limit. When the number of connected LPs increases the synchronization becomes a major restriction in the simulation. Therefore, synchronization should be minimized as much as possible. In this thesis the novel idea of an LP *neighborhood* is introduced to reduce the synchronization.

The idea of the neighborhood can be seen from Figure 8 and Figure 9. The original neighborhood of LP3 is presented in Figure 8. This neighborhood contains all connections from the physical model (see Figure 1 and Figure 2). Instead of connecting LP3 to all necessary LPs, LP3 can be connected only to the most significant LPs. When the connections are reduced to the most important ones, the LP neighborhood presented in Figure 9 is achieved.

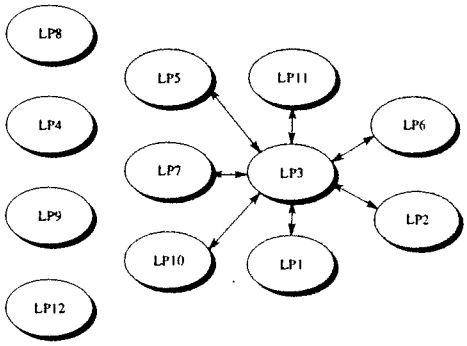

Figure 8. Original neighborhood of LP3.

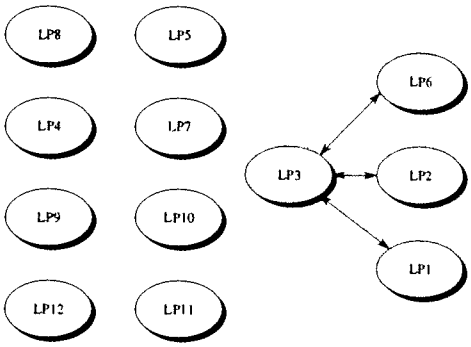

Figure 9. Reduced neighborhood of LP3.

By establishing a neighborhood for an LP, the number of its connections is reduced. As the number of logical channels decreases, the synchronization between LPs diminishes and the simulation proceeds with less constraints. The reduction of the synchronization is due to the lookahead used in the Chandy-Misra algorithm. The process times of the neighboring LPs must proceed within a lookahead, whereas the bounds are larger for other LPs. The neighborhood concept reduces the synchronization in the simulation but the synchronization to the LPs outside the neighborhood does not entirely disappear. The reason for this is that new neighborhoods extend the concept of synchronization outside the original neighborhood. Therefore, all LPs in the model are somewhat synchronized through the neighborhoods, but the synchronization diminishes as the distance between the LPs increases.

The selection criteria for the neighborhood are important as "direct" logical channels should be reserved for significant connections only. Several criteria, e.g. the volume and importance of messages in a logical channel, can be used for the selection. The suitable criteria depend heavily on the simulation application and it may be hard to select the optimal neighborhood for an LP based on a single criteria. In some cases the neighbors should be picked up one by one according to the application specific needs. For example, if only a limited number of messages is transmitted through a certain logical channel, then the LPs connected through that channel may have a weak dependency and they should not belong to each others' neighborhoods. On the other hand these messages may contain necessary information, which makes the channel vital for the simulation model and therefore those LPs should be included into each others' neighborhoods. Similar conclusions can be made for the high volume of messages. Thus, the connections are highly application dependent and a good knowledge of the application is needed for the optimal neighborhood selection.

The neighborhood concept reduces the synchronization efficiently by removing connections between the communicating LPs. Although the connections are removed, the communication need between LPs may still exist. In order to achieve as correct results as possible, messages from these non-connected LPs should be taken into account. Since the basic Chandy-Misra algorithm does not take events from a non-connected LP into account, a method for their transmission must be implemented. Two separate methods have been studied and are proposed by the author.

Message path

The Message path method is based on a closed path from a logical process through other LPs back to the source LP. Therefore, a non-connected LP may be reached by sending a message through this closed path. Figure 10 presents a sample network of logical processes and a possible path through the model. The different shades in this figure represent the division of LPs into four workstations as can be done in Di*worse*. In this case there are four workstations and each of them contains a group of LPs.

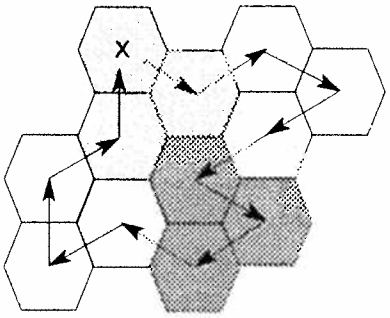

Figure 10. Message path from the transmitting LP to the others.

Message path is a simple method for the delivery of messages to non-connected LP(s) as it only requires the existing structures of the communication. The transmission path uses the same logical channels that are implemented for the neighborhood communication. However, this method is suitable only for those messages that have a small effect to the simulation, since all messages in this path are delayed in proportion to the length of the path. The closed path also causes a non-optimal network usage in the distributed environment as messages may be delivered several times between the same two workstations. In some cases it might be hard to find a closed path in the simulation model.

Broadcast messages

The Message path method successfully reduces the connections by using the neighborhood concept, but the drawback is the delay introduced by the path. The Broadcast messages method uses the clustered workstation structure of Di*worse* to reduce the delay between non-connected LPs. The delay of a message can be reduced by transmitting the message to all necessary LPs at the same time. This can be done by using a centralized transmission of messages. A message is delivered to all necessary workstations at the same time and the workstations then locally deliver the message to the corresponding LPs. Figure 11 presents a sample network of logical processes and the delivery of a message to all workstations and LPs. The thick arrows represent the message from one workstation to the others. The thin lines present the delivery of messages to the local LPs.

The Broadcast message method is more complicated than the Message path method, as separate queues are needed for the non-connected LPs. As new non-synchronous channels are added, the Chandy-Misra algorithm has to be modified to take the messages from the non-connected LPs into account. The implementation of the broadcast messages method can be done in several ways. The original channels can be divided into synchronous (for

connected LPs) and non-synchronous (for non-connected LPs) channels, or removed channels can be replaced by a single non-synchronous channel. In this thesis the latter approach has been used as it requires only minor changes to the implemented algorithm.

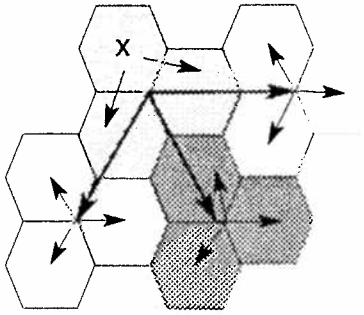

Figure 11. Broadcast messages method.

In the Broadcast messages method the overall delay of a message is smaller than in the case of message path but the modified algorithm should observe that in some rare cases the event times need to be corrected. The network usage in a distributed workstation environment is near optimal as the message is delivered only once to each workstation.

Accuracy of the simulation results

The neighborhood idea is based on the locality of the communication. For example in the cellular network application a minority of the channels take care of the majority of communication and thus most of the channels transfer mainly null messages. This disparity is corrected by cutting down the idle channels. This cutting does not come for free as the neighborhood is restricted and the synchronization is reduced at the expense of simulation accuracy. The effect of restrictions on the neighborhood need to be verified in order to preserve the desired accuracy in the simulation.

The basic Chandy-Misra algorithm and the lookahead property ensure that the simulation events occur in the correct order and the LPs are synchronized. However, the restriction of the neighborhood in the simulation model and the proposed communication method between non-connected LPs may have the result that an additional delay is introduced to the events. In the Message path method the delay increases as the message advances in the closed path and the lookahead is added to the event time in each LP. This delay inevitably creates inaccuracies to the results as events happen too late.

In the Broadcast message method simulation events suffer from a smaller delay but correction of event times may be needed due to the distance between the LPs. As the distance between the LPs in the simulation model increases, the dependency of the LPs decreases, which allows the LPs to proceed within a broader time difference. Therefore, non-connected LPs may proceed at different pace, as a result of which process times may differ by several lookaheads and some event times need correction.

Event times need to be adjusted whenever a non-connected LP with smaller process time sends a broadcast message to an LP that already has passed that time. In this case the

message time need to be increased to the current process time of the LP, i.e. rollback of events is not carried out. Even though event times need to be corrected in the Broadcast message method, the actual effect on the simulation is smaller than the effect of event delay in the Message path method. The difference of the inaccuracies in these methods increases as the size of the simulation model is increased.

3.5 Discussion

The performance of the distributed simulation environment can be improved by carefully considering the different aspects affecting the simulation. The main emphasis in our algorithmic modifications is on the consideration of the communication delay and its reduction. Most developments aim at the reduction of transmitted events, thus reducing the effect of a relatively slow communication network in our distributed simulation environment.

Null message cancellation has a significant role in reducing unnecessary synchronization messages. Through this technique the number of messages in the simulation was reduced to one tenth of the original amount (see Publication 2). Simulation time dropped with the same proportion. The proposed technique of simulating multiple messages in each simulation loop improved the simulation even more. Through this modification the results became more insensitive to the simulation load, i.e. the curves became almost linear (see Publication 2).

The simulation was further improved by considering the simulation application, i.e. cellular network. The proposed neighborhood concept and the reduced communication accelerated the simulation considerably with only a minor effect on the simulation accuracy. The Message path method was only slightly slower than the broadcast messages method but the simulation accuracy of the message path method was notably worse than in the case of broadcast messages [Por97]. In the broadcast messages method the simulation accuracy remained good even when the connections were reduced. Algorithmic speedups between 3.8 to 8.7 were achieved when compared to the original model (see Publication 4).

Chapter 4

Analytical Study of Distributed Simulation

Analytical methods can be used for studying the parallel properties of the simulation model or the application before actually implementing any parallel code. The critical path analysis presented by Berry and Jefferson in [Ber85] has been widely used for the analysis of different synchronization protocols. Supercritical properties of some optimistic algorithms have been shown by using this method [Gun94, Jef91, Sri95]. The following subsections present the basic ideas of the critical path analysis, an implementation of the existing ideas and a novel idea of how to use critical path analysis to reveal the effect of the simulation environment.

4.1 Critical path analysis

Critical path analysis is an analytical approach for solving the lower bound for the parallel execution time as several processors execute a problem. Critical path analysis is based on an event trace from the real simulation of the problem. The obtained trace is transformed into a directed event graph whose edges are weighted with real times representing computational and communicational delays. The weights need to be selected in such a way that they represent the timing characteristics of the execution hardware and the communication network. When the directed graph has been constructed a critical path algorithm can be applied to find out the longest weighted path in the graph. This logical path from the first event to the last event represents the lower bound for the simulation in the given hardware [Ber85]. This lower bound is valid mainly in the conservative methods as certain variants of the optimistic Time-Warp method are shown to be supercritical [Gun94, Sri95].

The creation of a directed graph for several processors is limited by two constraints, i.e. *predecessor* and *creator* relationships. These constraints restrict the parallel execution of events as the constraints need to be fulfilled before an event can be scheduled for the execution. The creator relationship represents a situation where an event e_1 causes an event e_2. Thus event e_1 (creator) naturally needs to be executed before event e_2. The predecessor relationship represents a situation where events e_1 and e_2 are scheduled to the same logical process and the timestamp of event e_1 is less than the timestamp of event e_2. Therefore, event e_1 is executed before event e_2. The earliest possible completion time τ_i, i.e. *critical time*, for event e_i can be calculated by using the critical times of its creator and predecessor as follows:

$$\tau_i = \max\{ \tau_{pred(i)}, \tau_{crea(i)} \} + \Delta\tau_i ,$$

where $\Delta\tau_i$ is the required execution time of event e_i and $\tau_{pred(i)}$ and $\tau_{crea(i)}$ are the critical times of the predecessor and creator respectively. The critical time of the last event in the critical

path represents the lower bound for the execution time of the conservative simulation. An example of critical path analysis is presented in Figure 12 and Table 1.

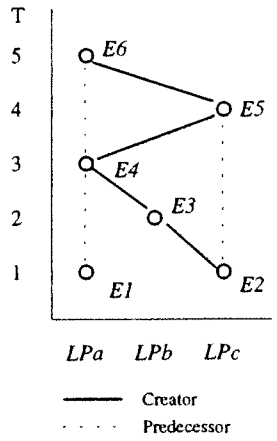

Figure 12. An example of a small simulation model.

Table 1. The earliest completion times of events in the simulation model.

E_i	$\Delta\tau_i$	pred(i)	crea(i)	τ_i
E1	5	-	-	5
E2	3	-	-	3
E3	2	-	*E2*	5
E4	1	*E1*	*E3*	6
E5	2	*E2*	*E4*	8
E6	1	*E4*	*E5*	9
Seq	14			

Figure 12 presents the logical processes (a, b and c), events (1 to 6) and their relations. The vertical axis shows the timestamps of the events and the horizontal the different LPs. Table 1 presents the events, their execution times, creator and predecessor events as well as the critical times τ_i of the events. The critical time of the whole simulation is the maximum of all critical times, in this case 9 units. The sequential simulation would require 14 units and thus the achieved speedup is $14/9 \approx 1.56$.

4.2 Revealing parallel properties

The critical time computing requires information about the events, their relations and execution times. This information can be easily obtained by collecting a trace from the actual simulation and by measuring the event execution times in a given environment. In this thesis the critical time calculation is divided into two classes, i.e. sequential and parallel, according to the method used for trace gathering. The method based on a trace from sequential simulation has been used by several authors. In this thesis another method that uses a trace from parallel simulation is proposed. The following subsections present the applicability of these two approaches.

Sequential event trace method

In this thesis the sequential event trace method has been used for computing the *parallel potential* of the application, i.e. the efficiency how well the increased number of processors can be utilized with the given application and the selected simulation model. A simple procedure was implemented to do this. This procedure follows the work done by Berry and Jefferson in [Ber85]. The phases in the procedure to assess the parallel potential of a simulation model are summarized in Figure 13.

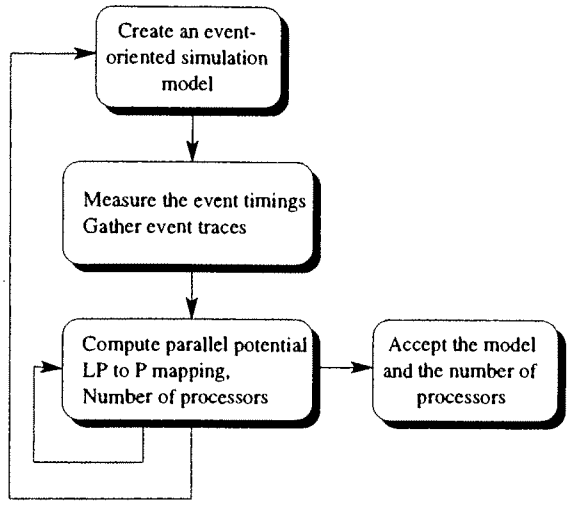

Figure 13. A simple procedure for calculating the parallel potential of the application.

In the beginning, a real system must be modeled by identifying the logical processes and events. The decisions made in this phase have a direct impact on the parallel potential of the model and therefore, the modeling is an iterative process. The second phase of the procedure consist of the gathering of event traces and the measuring of the event execution times in the execution environment. In this phase a simple sequential implementation of the simulator is sufficient. Lin has proposed a method that does not require separate trace collection [Lin92] but in this work the trace file was produced by a sequential simulation. Although the event execution times depend on the architecture, it is anticipated that in the broad scale, the computed parallel potential gives a good insight of the applicability of the model to other architectures as well. The third phase of the procedure is the computation of the parallel potential of the simulation application. The produced trace file and the measured event execution times are needed at this phase. For the computation, the user can specify the number of real processors (P) and the LP to P mapping algorithm. Three alternatives have been implemented for the mapping algorithm:

- fixed one-to-one mapping,
- fixed user defined mapping and
- dynamic mapping.

These mappings are illustrated in Figure 14.

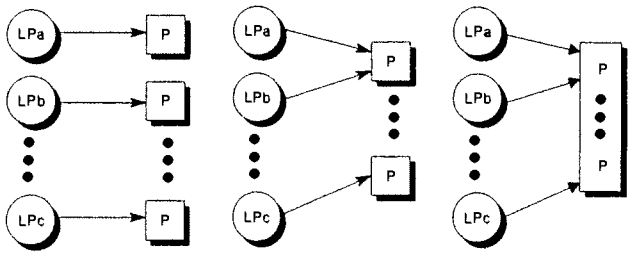

Figure 14. LP to P mappings.

The fixed one-to-one mapping executes every logical process in its own physical processor. This alternative produces directly the critical time of the simulation model, i.e. the selected simulation model cannot be executed faster. This is usually done first for any simulation model, since the achieved speedup gives quickly a rough estimate of the reasonable number of processors that should be allocated for the simulation model. Using as many processors as there are logical processes is rarely reasonable for a large number of logical processes, since often the model cannot fully utilize the available parallelism.

The next step is to search for a knee point, beyond which adding more processors does not significantly increase the speedup. In architectures where dynamic mapping of events to processors is not possible, e.g. distributed environments, the user may want to use the second alternative, the fixed user defined mapping. In this mapping the user defines which logical processes are to be executed in a given physical processor. The load should be divided equally to every physical process in order to get the best result. In all other respects, the third alternative, dynamic mapping is preferable. Whenever an event becomes eligible for execution, in dynamic mapping it is simulated by the processor that has been idle for the longest period.

Usually, only a few computations around the number of processors reported by the fixed one-to-one mapping is required to find out the suitable number of processors. If the speedup is still low and enough processors are available, another fixed LP to P mapping, if used, should be tried first. If the remapping does not help or the fixed user defined mapping is not used, a reconstruction of the LP structure of the simulation model may help.

Parallel event trace method

The use of the sequential trace in critical path analysis reveals the parallel properties of the application, the simulation algorithm, the selected simulation model as well as the simulation environment in environments where the communication cost is insignificant. The distributed simulation environment differs from the shared memory environment as the communication network has its own effect on the simulation. The use of the sequential trace is still advisable for the comparison of different simulation models. However, the use of the sequential trace in the distributed environment merely illustrates the parallel properties of the application, not the simulation algorithm or the environment. Therefore, in the distributed simulation environment the sequential trace is insufficient to capture all the aspects of parallel simulation. Both algorithmic and environmental properties can be analyzed by running the application in parallel and obtaining the trace from the parallel run. This requires parallel implementation of the program but allows the examination of the distributed environment. This novel idea has been proposed and implemented by the author of this thesis.

Our critical time calculations in the distributed environment were performed by using simulation traces and event execution times from the actual distributed simulations. Figure 15 presents the proposed procedure that can be used for obtaining critical time results for the given problem in a distributed environment. The proposed procedure captures application, algorithm and environment depending aspects.

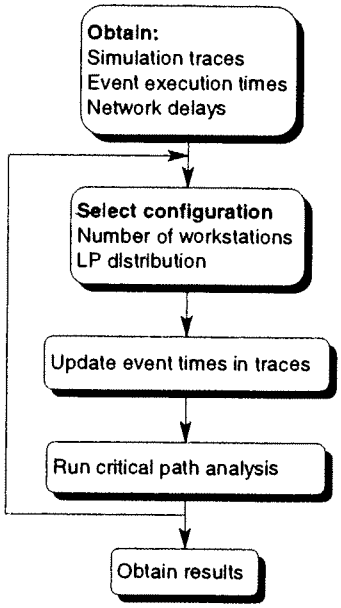

Figure 15. The procedure for obtaining the critical time of the application.

In the measurement phase the simulation traces and information concerning event execution times are gathered. Both the event processing times in a certain environment and network delays for each event need to be measured as the critical times depend on both of these aspects. When the traces and measurements are obtained the configuration, i.e. the desired number of workstations and the LP distribution, can be selected. It should be noticed that the user defined LP to P mapping need to be used as dynamic mapping is not reasonable in the distributed environment. The event execution times are updated to the traces according to the selected configuration. Critical path analysis is then performed by using these updated traces. Same traces can be used for different configurations by updating the configuration information to the traces, i.e. changing event execution times and distribution information. The proposed technique can be used for estimating the behavior of the simulation model with the given number of processors and LP distribution. The effect of different LP distributions can also be studied.

4.3 Discussion

Critical path analysis can be used for predicting the performance of the given simulation application, algorithm, simulation model or environment. If there is sufficient knowledge about the application's parallel potential with different simulation models, this analysis can be performed by using a simple sequential trace. If the properties of the algorithm or the environment are required, the sequential trace is insufficient and the proposed parallel technique needs to be used.

The parallel potential of the simulation application has been studied in Publication 5. For dynamic LP to P mapping the speedup curves show a clear upper bound for the number of useful processors. However, dynamic mapping is applicable mainly in shared memory environments. Static mapping corresponds to distributed architectures as LPs cannot be

dynamically moved from one LP to another due to high communication costs. With static mapping no clear knee point was found. According to the results, the application and the selected simulation model seem to have a moderate portion of parallel potential, i.e. a large number of processors can be utilized quite efficiently.

The effect of the distributed environment to the parallel potential has been studied in Publication 6. A user defined LP to P mapping was used instead of dynamic mapping. A knee point, where the network delay starts dominating the simulation, was expected to be seen in the results. This knee point was found and this proved that the selected application can be simulated on a distributed environment quite effectively. The achieved results also revealed that critical path analysis can be used as a fairly accurate performance prediction method.

Chapter 5

Conclusions

Parallel and distributed simulation is an expanding area of research as more and more complex problems need to be solved within reasonable time limits. A lot of work has been done during the last few decades in this field. In the 70's and 80's the research concentrated on the development of different parallel simulation methods and algorithms. In the 90's a larger portion of research has been dedicated to different applications and tools. This thesis presents a study that uses an old parallel simulation algorithm [Cha79b] and modifies it to be used efficiently in a distributed environment. Both existing and new approaches have been used in this process. The basic Chandy-Misra algorithm, the null message cancellation method, LP clustering and critical path analysis are examples of existing techniques used in this thesis. Multiple message simulation, neighborhood reduction and parallel trace based critical path analysis are the main ideas developed by the author. Through these methods the original simulator based on the use of shared memory was successfully implemented to a distributed environment. The work done in this thesis can be divided into three parts:

- implementation of the distributed simulation environment
- algorithmic improvements
- analysis of the implemented environment and the application

The distributed simulation environment Di*worse* is based on the use of a network of workstations. The workstations are used for the simulation in such a way that the communication through a relatively slow interconnection network is minimized. The structural changes made the environment viable for parallel and distributed simulation. Especially the LP clustering technique and the use of the Scheduler element reduced the additional overhead caused by the distribution. The poor parallel execution results presented in Publication 1 and Publication 2 were due to the inefficient structure of the simulation environment in the early stage of the research. This part of the work consists mainly of known techniques.

The Chandy-Misra parallel simulation algorithm was modified to be used in the distributed environment. As the amount of synchronization messages was quite high in this conservative algorithm, almost all the algorithmic improvements were directed to the minimization of the communication between the logical processes. The use of tthe proposed multiple message simulation method and an existing null message cancellation method improved the performance significantly. Together with structural changes the modified simulation algorithm achieved some speedup when compared to sequential simulation (see Publication 3). Through the neighborhood concept and careful consideration of the application even better results were achieved (see Publication 4).

The simulation application, the algorithm and the implemented environment were analyzed by using the well-known critical path analysis. The use of sequential event traces in the analysis showed that the used application has plenty of inherent parallel potential, so good speedup results can be expected. The effects of the algorithm and the environment were studied by applying critical path analysis to the same application but this time on a distributed environment. The proposed method of using parallel trace from the distributed environment confirmed the good parallel potential of the application. At the same time the analysis revealed the effects of the algorithm and the environment. Both algorithmic and structural modification were quite successful as good speedups were achieved. The effect of LP distribution to the simulation was also studied. LP distribution seems to have a significant role as it affects the amount of communication.

Although quite a few aspects were improved during the work, there is still lot to do. Static partitioning of the problem can be implemented easily but is hardly the best solution to the LP distribution. Therefore, load balancing techniques need to be considered in the context of the implemented environment. The effect of different networks, e.g. ATM, is still unclear. The accuracy of critical path analysis can be further improved and different applications should be considered in order to reveal the application dependent properties of the developed improvements. Even though a lot may be improved, the developed improvements have shown that the implemented distributed environment can be used efficiently for the simulation purposes, at least with the cellular network application.

Chapter 6

Summary of the Publications

This thesis consists of six publications on parallel and distributed simulation of cellular networks. The publications concentrate on the aspects of developing a distributed simulation environment by using a network of workstations. One of the articles has been published in an annual review publication and five in conference proceedings. Publications 1 - 4 concentrate on the development of a distributed simulation environment, whereas publications 5 and 6 focus on the analytical modeling of the simulation.

Publication 1 introduces the distributed workstation concept and the first implementation of the distributed simulation environment. In this implementation no optimizations are considered. Logical processes are realized as separate physical processes. The communication is based on the use of UDP connections between workstations. A four -cell GSM network model is used as an application on top of the simulation environment. The achieved results indicate that this first implementation of the distributed simulation environment cannot compete with sequential implementation.

Publication 2 presents improvements developed for the basic Chandy-Misra parallel simulation algorithm. The improvements are designed for the use of the implemented distributed workstation environment. The use of null message cancellation improves the algorithmic performance considerably and the novel idea for allowing simulation of several messages (SimL∞p algorithm) further improves the performance. Logical processes are realized as separate physical processes and the communication is based on the use of UDP connections between workstations. Multicasting technique is used for the improvement of the communication. Due to the poor performance of the basic Chandy-Misra algorithm only a two -cell GSM network is used for the experiments and comparisons. The achieved results indicate that the algorithmic optimizations improve the performance considerably. Although good improvements are achieved the sequential implementation cannot be beaten unless the structure of the simulation environment is changed.

Publication 3 concentrates on the parallel simulation of cellular network applications. Different aspects affecting the simulation of cellular systems are considered. The distributed simulation environment is used for the simulation of a 27 -cell GSM network application. In this publication the structure of the environment is improved in such a way that in each workstation only one physical process is used. Logical processes allocated for the workstation are implemented inside this process. Communication between workstations is implemented by using TCP instead of UDP. The achieved results indicate that the distributed simulation can beat the sequential approach.

Publication 4 presents a novel idea (LP neighborhood) for reducing the number of logical channels in the distributed simulation algorithm. By cutting down the amount of logical channels according to the neighborhood concept, the synchronization between logical processes can be reduced. The reduced synchronization can be observed as a decreased number of messages and more independent execution of logical processes. In this paper each workstation contains only one physical process and the communication between workstations is implemented by using TCP. A 27 -cell GSM network is used as the simulation application. The modifications presented in this paper improve the distributed simulation algorithm 3.8 to 8.7 times according to the offered load and the number of workstations. After the modifications the distributed simulation performs 1.8 to 3.9 times faster than the sequential implementation.

Publication 5 introduces a simple procedure for the parallel potential calculation of applications in shared memory systems. The calculation is based on the well-known concept of critical time of the simulation. The procedure requires a measurement of the event timings and a trace from the sequential simulation. Event timings and the trace are fed to a trace-driven simulator which calculates the critical time for the application. Critical times can then be used for estimating the necessary amount of processors. Parallel potential is calculated for a 27 -cell GSM network application. The results of this research give an overview of the parallel properties of the used GSM application. These simulation experiments were performed in a 2 -processor Convex machine. The results indicate that only a half of the maximum number of processors are really needed.

Publication 6 extends critical time calculation to the distributed simulation concept by introducing a novel idea of using parallel event traces. In addition to the trace and the event processing, the calculation of the critical times for the distributed environment requires information about the event delays in the transmission network. Event processing times can be easily measured from the sequential approach but the trace and network delays need to be obtained from a distributed simulation. The critical time analysis is used for solving the behavior of the application as several workstations are used. In this paper the critical time analysis is performed for a 27 -cell GSM network application. The results indicate that approximately a half of the maximum number of processors can be used efficiently in a distributed simulation environment.

Errata

In **Publication 1** in Figure 2 there is two arrows between LP1 and LP2 when only one is required.

In **Publication 6** Figure 4 reads: Measured processing times / event in SUN IPX environments (ms). It should be: Measured processing times / event in SUN IPX environments (μs).

Bibliography

[Ana97] Anastasiadis S. and Sevcik K.: Parallel Application Scheduling on Networks of Workstations, *Journal of Parallel and Distributed Computing*, 43, 1997, pp. 109-124.

[Aya93] Ayani R. and Berkman B.: Parallel Discrete Event Simulation on SIMD Computers, *Journal of Parallel and Distributed Computing*, 18, 1993, pp. 501-508.

[Bag95] Bagrodia R., Gerla M., Kleinrock L., Short J. and Tsai T.-C.: A Hierarchical Simulation Environment for Mobile Wireless Networks, *Proceedings of the 1995 Winter Simulation Conference*, pp. 563-570.

[Bal89] Bal H., Steiner J. and Tanenbaum A.: Programming Languages for Distributed Computing Systems, *ACM Computing Surveys*, Vol. 21, No. 3, September 1989, pp. 261-322.

[Ban93] Banawan S.: The Limited Benefits of Load Sharing in Multicomputer Systems, *Proceedings of the 26th Annual Simulation Symposium*, 1993, pp. 116-125.

[Ber85] Berry O. and Jefferson D.: Critical Path Analysis of Distributed Simulation, *Proceedings of the 1985 Conference on Distributed Simulation*, pp. 57-60.

[Bou91] Boukerche A. and Tropper C.: A Performance Analysis of Distributed Simulation with Clustered Processes, *Proceedings of the SCS Multiconference on Advances in Parallel and Distributed Simulation*, 1991, pp. 112-121.

[Cab97] Cabillic G. and Puaut I.: Stardust: An Environment for Parallel Programming on Networks of Heterogeneous Workstations, *Journal of Parallel and Distributed Computing*, 40, 1997, pp. 65-80.

[Cai90] Cai W. and Turner S.: An Algorithm for Distributed Discrete-Event Simulation - The "Carrier Null Message" Approach, *Proceedings of Distributed Simulation*, 1990, pp. 3-8.

[Car95] Carothers C., Fujimoto R. and Lin Y-B.: A Case Study in Simulating PCS Networks Using Time Warp , *Proceedings of the PADS'95*, pp. 87-94.

[Cav96] Cavitt D., Overstreet C. and Maly K.: A Performance Analysis Model for Distributed Simulations, *Proceedings of the 1996 Winter Simulation Conference*, pp. 629-636.

[Cha79a] Chandy K., Holmes V. and Misra J.: Distributed Simulation of Networks, *Computer Networks*, 3, 1979, pp. 105-113.

[Cha79b] Chandy K. and Misra J.: Distributed Simulation: A Case Study in Design and Verification of Distributed Programs, *IEEE Transactions on Software Engineering*, Vol. SE-5, No. 5, September 1979, pp. 440-452.

[Cha81] Chandy K. and Misra J.: Asynchronous Distributed Simulation via a Sequence of Parallel Computations, *Communications of the ACM*, Vol. 24, No. 11, April 1981, pp. 198-206.

[Cha89] Chandy K. and Sherman R.: The conditional event approach to distributed simulation, *Proceedings of the Distributed Simulation 1989*, 1989, pp. 93-99.

[Che98] Chen Y-A. and Bagrodia R.: Shared Memory Implementation of a Parallel Switch-Level Circuit Simulator, *Proceedings of the PADS'98*, 1998, pp. 134-141.

[Cle96] Cleary J. and Tsai J-J.: Conservative Parallel Simulation of ATM Networks, *Proceedings of the PADS'96*, 1996, pp. 30-38.

[Cub94] Cubaud P., Pekergin N. and Taleb H.: OPPSI: A Performance Prediction Tool for Parallel Systems, *Proceedings of the European Simulation Symposium 1994*, 1994, pp. 210-214.

[Dav90] Davis N., Mannix D., Shaw W. and Hartrum T.: Distributed Discrete-Event Simulation Using Null Message Algorithms on Hypercube Architectures, *Journal of Parallel and Distributed Computing*, 8, 1990, pp. 349-357.

[DeV90] De Vries R.: Reducing Null Messages in Misra's Distributed Discrete Event Simulation Method, *IEEE Transactions on Software Engineering*, Vol. 16, No. 1, January 1990, pp. 82-91.

[Dia96] Diab H. and Tabbara H.: Performance Factors in Parallel Programs, *International Journal in Computer Simulation*, 6, 1996, pp. 427-447.

[Dri95] Driscoll M. and Daasch W.: Accurate Predictions of Parallel Program Execution Time, *Journal of Parallel and Distributed Computing*, 25, 1995, pp. 16-30.

[Du97] Du X. and Zhang X.: Coordinating Parallel Processes on Networks of Workstations, *Journal of Parallel and Distributed Computing*, 46, 1997, pp. 125-135.

[Dun90] Duncan R.: A Survey of Parallel Computer Architectures, Survey & Tutorial series, *Computer*, February 1990, pp. 5-16.

[Fer94] Ferscha A. and Tripathi S.: Parallel and Distributed Simulation of Discrete Event Systems, *CS-TR-3336*, University of Maryland, August, 1994, ftp://ftp.cs.umd.edu/pub/papers/TRs/3336.ps.Z

[Fer96] Fercha A. and Johnson J.: A Testbed for Parallel Simulation Performance Prediction, *Proceedings of the 1996 Winter Simulation Conference*, 1996, pp. 637-644.

[Fos95] Foster I.: *Designing and building parallel programs, Concepts and Tools for Parallel Software Engineering*, Addison-Wesley, 1995.

[Fuj89] Fujimoto R.: Performance Measurements of Distributed Simulation Strategies, *Transactions of the Society for Computer Simulation*, 1989, pp. 89-132.

[Fuj90] Fujimoto R.: Parallel Discrete Event Simulation, *Communications of the ACM*, Vol. 33, No. 10, 1990, pp. 30-53.

[Gre94] Greenberg A., Lubachevsky B., Nicol D. and Wright P.: Efficient Massively Parallel Simulation of Dynamic Channel Assignment Schemes for Wireless Cellular Communications, *Proceedings of the PADS'94*, pp. 187-194.

[Gun94] Gunter M.: Understanding Supercritical Speedup, *Proceedings of the PADS'94*, pp. 81-87.

[Har93a] Harju J. and Salmi M.: Application of Conservative Parallel Simulation to the GSM Radio Telephone Performance Analysis, *Research Report 36*, Lappeenranta University of Technology, 1993, ISBN 951-763-772-1.

[Har93b] Harju J., Salmi M. and Porras J.: Parallel Simulation of Mobile Communication Systems, *Poster presentation in the 7th Nordic Symposium on Computer Science*, 1993, p. 47.

[Har93c] Harju J. and Salmi M.: Parallel Simulation of the GSM Mobile Communication Network on Minisupercomputer, *Proceedings of the 1993 Simulation Multiconference on High Performance Computing*, 1993, pp. 182-187.

[Har94] Harju J., Salmi M. and Porras J.: Parallel Discrete-Event Simulation in Supercomputers, *CSC News*, Vol. 6, No. 1, 1994, pp. 3-5.

[Hut98a] Huttunen P., Porras J., Ikonen J. and Sipilä K.: Using Cray T3E for the parallel calculation of cellular radio coverage, *Proceedings of the Eurosim'98*, 1998, pp. 27-32.

[Hut98b] Huttunen P., Porras J., Ikonen J. and Sipilä K.: Parallelization of cellular radio coverage calculation, *Proceedings of the 10th European Simulation Symposium*, 1998, to appear.

[Hyb98] Hybinette M. and Fujimoto R.: Dynamic Virtual Logical Processes, *Proceedings of the PADS'98*, 1998, pp. 100-107.

[Jef85] Jefferson D.: Virtual Time, *ACM Transactions on Programming Languages and Systems*, Vol. 7, No. 3, 1985, pp. 404-425.

[Jef91] Jefferson D. and Reiher P.: Supercritical Speedup, *Proceedings of 24th Annual Simulation Symposium*, pp. 159-168.

[Kau87] Kaudel F.: A Literature Survey on Distributed Discrete Event Simulation, *Simuletter* 18, Feb. 1987, pp. 11-21.

[Kel96] Keller J., Rauber T. and Rederlechner B.: Conservative Circuit Simulation on Shared-Memory Multiprocessors, *Proceedings of the PADS'96*, pp. 126-134.

[Kom96] Komatsu T., Nakamura Y. and Nose J.: Simulator for the Evaluation of Distributed Network-System Performance, *Proceedings of the 1996 Winter Simulation Conference*, 1996, pp. 1222-1229.

[Kon91] Konas P. and Yew P-C.: Parallel Discrete Event Simulation on Shared-Memory Multiprocessors, *IEEE*, 0272-4715/91/0000/0134, 1991, pp. 134-148.

[Kum94] Kumar V., Grama A., Gupta A. and Karypis G.: *Introduction to Parallel Computing*, Benjamin/Cummings, Publishing Company, 1994, ISBN 0-8053-3170-0.

[Lin90] Lin Y-B. and Lazowska E.: Exploiting Lookahead in Parallel Simulation, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 1, No. 4, October 1990, pp. 457-469.

[Lin92] Lin Y-B.: Parallelism Analyzers for Parallel Discrete Event Simulation, *ACM Transactions on Modeling and Computer Simulation*, Vol. 2, No. 3., July 1992, pp. 239-264.

[Lin93] Lin Y-B.: Special Issue on Parallel Discrete Event Simulation, Guest Editor's Introduction, *Journal of Parallel and Distributed Computing*, 1993, pp. 391-393.

[Liu96] Liu W., Chiang C., Wu H., Jha V., Gerla M. and Bagrodia R.: Parallel Simulation Environment for Mobile Wireless Networks, *Proceedings of the 1996 Winter Simulation Conference*, pp. 605-612.

[Mal95] Malloy B. and Montroy A.: A parallel distributed simulation of a large-scale PCS network: keeping secrets, *Proceedings of the 1995 Winter Simulation Conference*, pp. 571-578.

[Mey97] Meyer T., Davis J. and Davidson J.: Analysis of Load Average and its Relationship to Program Run Time on Networks of Workstations, *Journal of Parallel and Distributed Computing*, 44, 1997, pp. 141-146.

[Mis86] Misra J.: Distributed Discrete-Event Simulation, *Computing Surveys*, Vol 18, No. 1, March 1986, pp. 39-65.

[Nic93] Nicol D.: The Cost of Conservative Synchronization in Parallel Discrete Event Simulation, *Journal of Association for Computing Machinery*, Vol. 40, No. 2, April 1993, pp.304-333.

[Nic94] Nicol D. and Fujimoto R.: Parallel Simulation Today, *Annals of Operations Research*, Vol. 53, 1994, pp. 249-286.

[Nic95] Nicol D. and Heidelberger P.: On Extending Parallelism to Serial Simulators, *Proceedings of the PADS'95*, 1995, pp. 60-67.

[Nic96] Nicol D. and Heidelberger P.: On Extending More Parallelism to Serial Simulators, *Proceedings of the PADS'96*, 1996, pp. 202-205.

[Pan97] Panda D. and Ni L.: Special Issue on Workstation Clusters and Network-Based computing, Guest Editors' Introduction, *Journal of parallel and distributed computing*, 40, pp. 1-3.

[Pan98] Panchal J., Kelly O., Lai J., Mandayam N., Ogielski A. and Yates R.: WiPPET, A Virtual Testbed for Parallel Simulations of Wireless Networks, *Proceedings of the PADS'98*, 1998, pp. 162-169.

[Pea79] Peacock J., Wong J. and Manning E.: Distributed Simulation Using a Network of Processors, *Computer Networks*, 3, 1979, pp. 44-56.

[Per98] Perrone L. and Nicol D.: Rapid Simulation of Wireless Systems, *Proceedings of the PADS'98*, 1998, pp. 170-177.

[Pha98] Pham C., Brunst H. and Fdida S.: Conservative Simulation of Load-Balanced Routing in a Large ATM Network Model, *Proceedings of the PADS'98*, 1998, pp. 142-143.

[Por97] Porras J., Hara V., Harju J. and Ikonen J.: Reducing the Number of Logical Channels in the Chandy-Misra Algorithm, *Proceedings of the 9th European Simulation Symposium* (ESS'97), 1997, pp. 252-256.

[Pre91] Preiss B., Loucks W., MacIntyre I. and Field J.: Null Message Cancellation in Conservative Distributed Simulation, *Proceedings of the SCS Multiconference on Advances in Parallel and Distributed Simulation*, 1991, pp. 33-38.

[Ree88] Reed D., Malony A. and McCredie B.: Parallel Discrete Event Simulation Using Shared Memory, *IEEE Transactions on Software Engineering*, Vol. 14, No.4, April 1988, pp. 541-553.

[Rön94] Rönngren R., Rajaei H., Popescu A., Liljenstam M., Ismailov Y. and Ayani R.: Parallel Simulation of a High Speed LAN, *Proceedings of the PADS'94*, 1994, pp. 132-138.

[Sal93a] Salmi M, Harju J. and Porras J.: Parallel Simulation and Its Application to Mobile Network Design, *Proceedings of the Second Summer School on Telecommunications*, 1993, pp. 575-579.

[Sal93b] Salmi M., Harju J. and Porras J.: A Chandy-Misra Parallel Simulation Environment for the Simulation of GSM Mobile Communication Network, *Proceedings of the 1993 European Simulation Symposium*, 1993, pp. 575-579.

[Sch95] Schlagenhaft R., Ruhwandl M., Sporrer C. and Bauer H.: Dynamic Load Balancing of a Multi-Cluster Simulation on a Network of Workstations, *Proceedings of the PADS'95*, 1995, pp. 175-180.

[Shi92] Shirazi B. and Hurson A.: Special Issue on Scheduling and Load Balancing, Guest Editor's Introduction, *Journal of Parallel and Distributed Computing*, 16, 1992, pp. 271-275.

[Sim95] Simonovich D.: Merging Processes in Parallel Discrete-Event Simulation, *Proceedings of European Simulation Symposium 1995*, 1995, pp. 61-64.

[Sri95] Srinivasan S. and Reynolds P.: Super-criticality revisited, *Proceedings of the PADS'95*, 1995, pp. 130-136.

[Su89] Su W-K. and Seitz C.: Variants of the Chandy-Misra-Bryant distributed discrete-event simulation algorithm, *Proceedings of Distributed Simulation 1989*, 1989, pp. 38-43.

[Tac98] Tacic I. and Fujimoto R.: Synchronized Data Distribution Management in Distributed Simulations, *Proceedings of the PADS'98*, 1998, pp. 108-115.

[Tay95] Taylor S., Fatin F. and Delaitre T.: Estimating the Benefit of the Parallelisation of Discrete Event Simulation, *Proceedings of the 1995 Winter Simulation Conference*, 1995, pp. 674-681.

[Teo94] Teo Y. and Tay S.: Conservative Parallel Simulation on a Network of Workstations, *Proceedings of the European Simulation Symposium 1994*, 1994, pp. 158-163.

[Tur98] Turner S., Cai W., Lim C., Low Y., Hsu W. and Huang S.: A Methodology for Automating the Parallelization of Manufacturing Simulations, *Proceedings of the PADS'98*, 1998, pp. 126-133.

[Wag89a] Wagner D. and Lazowska E.: Parallel Simulation of Queueing: Limitations and Potentials, *Performance Evaluation Review*, Vol. 17, No. 1, May 1989, pp. 146-155.

[Wag89b] Wagner D., Lazowska E. and Bershad B.: Techniques for efficient shared-memory parallel simulation, *Proceedings of Distributed Simulation 1989*, 1989, pp. 29-37.

[Wag91] Wagner D.: Algorithmic optimizations of conservative parallel simulations, *Proceedings of SCS multiconference on Advances in Parallel and Distributed Simulation*, 1991, pp. 25-32.

[Won96] Wong Y-C. and Hwang S-Y.: Investigation of Parallel Simulation, *International Journal in Computer Simulation*, 6, 1996, pp. 175-205.

[Woo94] Wood K. and Turner S.: A generalized carrier-null method for conservative parallel simulation, *Proceedings of the PADS'94*, 1994, pp. 50-57.

[Zak97] Zaki M., Li W. and Parthasarathy S.: Costomized Dynamic Load Balancing for a Network of Workstations, *Journal of Parallel and Distributed Computing*, 43, 1997, pp. 156-162.

[Zen98] Zeng X., Bagrodia R. and Gerla M.: GloMoSim: A Library for Parallel Simulation of Large-scale Wireless Networks, *Proceedings of PADS'98*, 1998, pp. 154-161.

[Zho97] Zhou W.: A tool for layered analysing and understanding of distributed programs, *Computer Communications*, 20, 1997, pp. 385-392.