Proceedings of the 1994 Winter Simulation Conference
ed. J. D. Tew, S. Manivannan, D. A. Sadowski, and A. F. Seila

DISTRIBUTED STOCHASTIC DISCRETE-EVENT SIMULATION IN PARALLEL TIME STREAMS

Krzysztof Pawlikowski
Victor W. C.Yau
Don McNickle

University of Canterbury
Christchurch
NEW ZEALAND

ABSTRACT

Quantitative stochastic simulation suffers from the fact that sound simulation studies require very long runlength to obtain the results with sufficient accuracy. In this paper we look at traditional approaches to distributed quantitative stochastic simulation and propose a new scenario, Multiple Replications in Parallel Time Streams (MRIP), that solves the problem in an efficient way. An implementation of MRIP in a simulation package AKAROA is also described. AKAROA accepts ordinary (non-parallel) simulation models and creates automatically the environment required for running MRIP on workstations of a local area network. Presented results show that MRIP offers linear speedup of simulation. Limitations of this scenario for running distributed quantitative stochastic simulation are also discussed.

1. INTRODUCTION

An inherent problem of stochastic simulation is that simulation of even moderately complex models can be computationally intensive and require very long simulation runs. The obvious solution is to speed up simulation by executing it on a multiprocessor or distributed computer system. Traditionally, distributed or parallel stochastic simulation has meant *Single Replication in Parallel* (SRIP), based on many processors cooperating in executing a single replication of a simulated system. An alternative scenario is to run *Multiple Replications in Parallel* (MRIP), with processors engaged into running their own replications of the simulated system but cooperating with central analyzers (one central analyzer for each performance measure analyzed) that are responsible for observing the stopping criteria of the simulation.

Research in distributed and parallel simulation has been almost entirely focused on SRIP. In this scenario, a simulation process and/or simulation model is partitioned between a number of processors. When a simulation process is distributed then this distribution is done at a functional level, and the logical topology of interprocessor connections may reflect different functional elements of the simulation (event set processing, input/output processing, etc.); see eg. (Biles, Daniels and O'Donnell 1985). It is obvious that this method cannot offer a substantial speedup in itself, since the degree of such distributiveness is limited.

The second option for SRIP is to partition a given simulation model into a set of submodels to be simulated at different processors, of tightly or loosely coupled multiprocessor systems. The processors responsible for running processes related with different simulation submodels occasionally have to synchronise the advance of simulated processes. Many different methods have been proposed to achieve such a synchronisation; see e.g. survey by Fujimoto (1990). Generally speaking, it is achieved by exchanging timestamped messages between participating processors. Reasonable speedup is possible, provided that a given simulation model is highly decomposable. Unfortunately, this feature is not frequently observed in practise, thus the efficiency of this approach is strongly application-dependent (Wagner and Lazowska 1989).

MRIP has focused relatively little attention [see papers by Heidelberger (1986), (1988), Glynn and Heidelberger (1991), Pawlikowski and Yau (1992), Rego and Sunderam (1991), (1992), Sunderam and Rego (1991), and Yau and Pawlikowski (1993)], despite that, as our experience has shown, it is an attractive alternative scenario for quantitative stochastic simulation that potentially offers good speedup, linear with the number of processors involved. Also, considering statistical properties of results when applying SRIP and MRIP in steady-state simulation, it is possible to show that the latter scenario is more efficient than the former, in the sense of the mean squared error of final estimates, if the problem of the initialization bias is effectively solved (Heidelberger 1986).

In this paper we report results of our research project on developing a fully automated version of MRIP and its implementation in AKAROA (a user-friendly package for running distributed quantitative stochastic simulation), started in 1991. It solves both main problems of such a simulation, by applying fully automated control of

accuracy of the final results (principles of which are summarised in Section 2), and fully automated parallelisation of simulation for concurrent execution on many procesors, see Section 3. The potentials and limitations of our version of MRIP, together with results of experimental studies of AKAROA are reported in Section 4. While AKAROA has many features in common with EcliPse discussed by Rego and Sunderam (1991), (1992), and Sunderam and Rego (1991), it creates a very different environment for running simulation. A user, having prepared a sequential simulation program has only to declare the required level of precision of the final results, e.g. 5%, and the maximum possible length of the simulation run (say, no more than 10 000 000 observations to be collected). All other decisions and functions are transparent for users.

Without a loss of generality we discuss simulation models based on queuing networks, and the application of quantitative stochastic simulation for studying performance for simulated systems in steady-state.

2 PRELIMINARIES: STATISTICAL ASPECTS OF NON-DISTRIBUTED STEADY-STATE SIMULATION

Any performance evaluation studies of systems based on quantitative stochastic simulation should include proper statistical analysis of output data. To control the precision of steady-state estimators, the final estimate of an analysed parameter Θ should be determined together with its confidence interval $(\theta-\Delta_1, \theta+\Delta_2)$, at a given confidence level $1-\alpha$, i.e. by $P(\theta - \Delta_1 \le \Theta \le \theta + \Delta_2)=1-\alpha$, where θ is the final estimate of Θ, and $(\Delta_2+\Delta_1)$ is the width of the confidence interval. The precision of estimates can be then measured by $\varepsilon = 0.5\ (\Delta_2+\Delta_1)/\theta$, known as the relative precision. In the context of quantitative steady state simulation, the precision of the final results can be controlled if it is sequentially checked at consecutive checkpoints, and compared with the (worst) acceptable level of precision, ε_{max}. The simulation is stopped at a given checkpoint if the stopping criterion (the current value of relative precision ε being not greater than ε_{max}, for given ε_{max}, $0<\varepsilon_{max}<1$) is satisfied for the first time.

This approach can be easily applied when analyzing performance measures using cumulative estimators. The simplest performance measure of this type is the sample mean μ_x, that, for a given sequence of observations x_1, x_2, ..., x_n, is estimated by the arithmetic average $\theta = \bar{X}(n) = (1/n)(x_1+x_2+...+x_n)$. In this case, by the Central Limit Theorem, $\Delta_1 = \Delta_2 = t_{n-1,1-\alpha/2}\ \hat{\sigma}[\bar{X}(n)]$, where $\hat{\sigma}[\bar{X}(n)]$ is the estimator of standard deviation of $\bar{X}(n)$ and $t_{n-1,1-\alpha/2}$ is the $(1-0.5\alpha)$ quantile of Student t-distribution. The main analytical problem is to get a reliable estimate of $\sigma^2[\bar{X}(n)]$, since the classical formulae require that collected observations $x_1, x_2, ..., x_n$ are realizations of independent and normally (or at least identically) distributed random variables $X_1, X_2, ..., X_n$. Unfortunately, observations collected during typical stochastic simulations are neither independent or identically distributed, but usually highly correlated.

A number of techniques for accurate estimation of $\sigma^2[\bar{X}(n)]$ have been proposed for non-distributed sequential simulation; see surveys by Law (1983) and Pawlikowski (1990). Our studies of these techniques (Pawlikowski and Yau 1991) led us to selection of SA/HW [the method based on spectral analysis, in its version proposed by Heidelberger and Welch (1981)] as the method of analysis of $\sigma^2[\bar{X}(n)]$ in fully automated quantitative steady-state simulations. SA/HW produces reliable estimates in the sense of their coverage (correspondence between the theoretical and experimental confidence levels) provided that simulated systems are not too heavy loaded, ie. are utilized no more than 90%. This is the common weakness of all techniques developed for automated analysis of $\sigma^2[\bar{X}(n)]$.

SA/HW requires that collected observations belong to stationary time series, thus observations collected during initial transient periods of analysed processes should be discarded. For detecting the length of initial transient period one can use a sequential stationarity test such as those surveyed in (Pawlikowski 1990) or (Stacey, Pawlikowski and McNickle 1993). Under SA/HW only one (long) simulation run is executed at each setting of input variables. A useful practical feature of this technique is that it can work with reduced data sets. During the whole course of a single simulation, we can work with a fixed number of (aggregated) output data points, being batch means calculated over batches that have their size increased as new observations are collected.

3 DISTRIBUTED QUANTITATIVE STOCHASTIC SIMULATION

Traditionally, applications of distributed, parallel processing in the area of stochastic simulation have focused on speeding up execution of single replications of simulation models. Another possible approach is to speed up such simulation experiments by generating statistical data in parallel, i.e. running replications of the simulated system in parallel, on many processors, under control of a global analyzer(s) responsible for analysing submitted data and detecting when the stopping condition of simulation is satisfied. Here, these two scenarios of distributed quantitative stochastic simulation are called Single Replication in Parallel (SRIP) and Multiple Replications in Parallel (MRIP), respectively.

3.1 Single Replication in Parallel

This can be achieved at a functional level of simulation, or it can be done by distributing the simulation model. These two solutions we call here briefly as *SRIP.F* and *SRIP.M*, respectively. In the former, the logical topology of interprocessor connections reflects different functional elements of the simulation and such activities as random variate generation, event list manipulation, and statistical analysis of output data are performed at separate processors. But the extent of functional distributiveness of any simulation is generally not significant, cf. (Comfort and Miller 1981), (Briner 1988). Let us also note that the fine granularity of decomposition of support functions needed for their parallel execution necessitates frequent communication among subprocesses. Apart from consuming processor power, interprocess communication puts limitations on multiprocessor architectures that can be used in such applications. Communication between processes on different processors may substantially increase total traffic on the time-shared bus and/or multiple-bus of multiprocessor systems to such extent that it becomes the bottleneck, increasing the amount of time the processors are blocked waiting for access to a common memory module. Even if the number of shared buses were increased, contention for memory (processors queue for accessing common memory module(s)) can create another bottleneck limiting effective processing power. Thus, one should not expect that this strategy allows to achieve a significant speedup (Burk 1990). Because of that, SRIP.F is not satisfactory efficient to be used on its own.

In SRIP.M, speedup is achieved by executing (interdependent) parts of the simulation model in parallel, at different processors. It is done by abandoning the concept of shared objects, such as the global simulation clock and event list, and using a synchronisation algorithm instead, to ensure that causality of events is maintained. The synchronisation of parallel (sub)streams of events simulated at different processors is achieved by exchanging time-stamped messages, in an attempt to protect against causality errors. For this purpose, either an *optimistic* or *conservative* synchronisation algorithm can be used; see (Fujimoto 1990).

There are natural weaknesses and limitations on efficiency of SRIP.M. Firstly, it has been shown that a high structural parallelism of simulation models does not imply similar high parallelism in the simulation of that model. For example, Wagner and Lazowska (1989) showed that the maximum speedup achievable when simulating a simple computer system (a CPU, two I/O devices and N terminals forming a closed queueing network) cannot be greater than 3.7. On the other hand, there are reports of many successful applications of distributed stochastic simulation executed in SRIP.M scenario, using either optimistic and conservative synchronisation algorithms. For example, a speedup as high as 1900 was demonstrated on a 16384 processor Connection Machine (Lubachevsky 1989). Generally, an experienced simulator should be able to obtain a good speedup, provided that a given simulation model is highly decomposable. Unfortunately, this feature is rarely observed in simulation practise. Thus, the success is strongly application-related. And, if deadlocks and causality errors occur too often, or mechanisms for protection against them are too complicated, the resulting simulation can be even slower then non-distributed one. This scenario of distributed stochastic simulation is also generally not well suited for running on distributed computer systems, such as networks of workstations, since there are substantial costs connected with interprocess communication. SRIP.M is also not fault-tolerant. If one processor or workstation running a sub-task fails, then the simulation fails too, due to causality between subtasks.

3.2 Multiple Replications in Parallel (MRIP)

In the light of these restrictions and limitations of SRIP, we have recognized that the duration of quantitative stochastic simulation directly depends on the time needed for collecting the required number of observations. If a few performance measures are studied during one simulation experiment, then the simulation is finished when all sequences of observations (one sequence for each performance measure) contain sufficiently many data items. Thus, a simple way for increasing the rate at which observations are generated is to produce them in parallel time streams, i.e. to run statistically different replications on many processors, using the same simulation model. One can view these simulation replications run at different processors as *simulation engines* working in a team and producing samples of output data (one sample per each performance measure). Observations generated by different simulation engines, but representing values of the same performance measure, are submitted to the global analyser responsible for analysing this performance measure. The current precision of results are checked at consecutive checkpoints. The analysis of each performance measure is then continued until its stopping condition is not satisfied. All simulation engines run as long as the analyses of all performance measures is finished. At that instant of time all simulation engines are stopped and global analysers produce the final results.

Distributed simulation in MRIP can be carried on with any simulation model, either on multiprocessor computers or multicomputer networks. Very little is know about estimators, even about estimators of sample mean, that could be applied in MRIP. Parallel versions of the method of Independent Replications, in the context of non-steady state simulation were analysed by Heidelberger (1988) and Glynn and Heidelberger (1991). It was shown that an extreme care has to be taken when selecting estimators since some obvious choices were shown to guarantee convergence to the wrong value when the number of processors increases. On the other

hand, studying properties of MRIP as the scenario for running steady-state simulation, it was shown that MRIP can be more efficient that SRIP.M if the problem of initial transient is effectively solved Heidelberger (1986).

From our previous experience in non-distributed stochastic simulation, we have decided to adopt the method of SA/HW (discussed in Section 2) also in distributed simulation, in the MRIP scenario. Our proposal is a parallel generalization of SA/HW, *Spectral Analysis in Parallel Time Streams* (SA-PTS). According to SA-PTS, P logically equivalent instances of a simulation model are launched at P processors at the beginning of the simulation. Each instance is run in a parallel time-stream, using different sequence of (pseudo)-random numbers. At the beginning, stationarity tests are applied locally within each replication, to determine the onset of steady state conditions in each time-stream separately, and the sequential SA/HW method is used to estimate the variance of local estimates at consecutive checkpoints. At each checkpoint the current local estimate and its variance are sent to the global analyser which computes the current value of the global estimate and its precision. Thus, when the simulation engine i reaches its check point j, it sends a message containing the triple $\{n_j, \bar{X}_i(n_j), V_{ij}= \hat{\sigma}^2[\bar{X}_i(n_j)\}$, number of observations, the mean and its variance, to the global analyser responsible for analysis of $\bar{X}$. Thus, the global precision of each estimator is analysed following partially ordered sequence of checkpoints (checkpoints associated with the same simulation engine are ordered in time, but we get a randomly ordered sequence of checkpoints from different processors as they followed by a given global analyser). If P processors are used in a given simulation, each time when the global analyser is active it can use up to P partial estimates of variance, $\{V_{1j_1}, V_{2j_2}, ..., V_{Pj_P}\}$, submitted from independent replications of the simulated process that reached the checkpoints $j_1, j_2, ...,$ and j_P, respectively. When p of P processors have reached at least the first checkpoint of the process they simulate, the pooled mean is estimated over $k= n_{1j1}+n_{2j2}+...+n_{pjp}$ observations, i.e. using n_{1j1} data collected by processor 1 at its checkpoint j_1, n_{2j2} data collected by processor 2 at its checkpoint j_2, etc. Natural estimators with good statistical properties to use for the mean value of the response and its variance are

$$\bar{\bar{X}}(k)=\frac{n_{1j1}\bar{X}(n_{1j1})+..+n_{pjp}\bar{X}(n_{pjp})}{n_{1j1}+...+n_{pjp}}, \text{ and}$$

$$\hat{\sigma}^2_{SAPTS}[\bar{\bar{X}}(k)]=\frac{(n_{1j1})^2V_{1j1}+..+(n_{pjp})^2V_{pjp}}{k^2}$$

Let us note that for determining the confidence interval for the sample mean μ_x, one has to know the probability distribution of $(\bar{\bar{X}}(k)-\mu_x)/\hat{\sigma}_{SAPTS}[\bar{\bar{X}}(k)]$. This is approximately governed by Student t-distribution with p times d degrees of freedom, where d equals the number of degrees of freedom of t-statistic coming from one replication. Let us note the obvious fault-tolerance of MRIP regarding simulation engines. Sudden loss of one or more processors running simulaiton engines is not catastrophic as long as at least one simulation engine remains able to continue submitting data to the global analyser(s).

At this stage no theoretical studies of SA-PTS are available. But our experimental results show that it produces good estimates in the sense of experimental converage of the final confidence intervals. SA-PTS has been implemented in AKAROA, our simulation package for rapid modeling, automatic generation of multiple processes and process control for concurrent stochastic simulation in MRIP scenario.

3.3 An Implementation of MRIP in AKAROA

AKAROA (a simulation package for automatic generation and control of processes for parallel stochastic simulation) accepts ordinary (non-parallel) simulation programs, and fully automatically creates the environment required for running MRIP on workstations of a local area network. Our main considerations when selecting a development language and designing programming interface were simplicity, space and code efficiency, as well as compatibility with existing sequential simulation programs. Recognizing the naturality of the object-oriented approach in constructing simulation models by means of hierarchically encapsulated classes of objects, AKAROA is written in C++. A user of AKAROA is required to add only one extra line of code to his/her sequential simulation program before AKAROA transparently parallelizes it. Thus, users do not even need to be aware of the existence of multiple (parallel) simulation engines and control processes during simulation, since their creation, location (machine and port addresses), cooperation, and inter-machine interprocess communication, are hidden from users. AKAROA consists of three modules: *Control*, responsible for controlling simulation run-time and analysis of output data collected during MRIP-type steady-state simulation; *Parallel Simulation Manager* (PSM), responsible for automatic initialization of parallel simulation processes, process management and interprocess communication; and *Build*, a module which can be used for speeding up construction of typical simulation models.

Sequential precision control services are arranged by declaring an object for output data analysis. Its member function responsible for precision control is later called whenever a new observation is recorded. The function accepts the value of a new observation as a parameter and returns one of two values that either orders the simulation to be continued (desired precision of estimates has not been achieved) or to be terminated (all estimates have

reached the required level of precision). Such implementation of MRIP is semantically identical to a normal non-distributed simulation; only the type of object that needs to be declared is different. The syntax for object declaration and calls of object's member functions are also identical to those in the non-parallel case. Internal binding of simulation processes to various control processes is performed dynamically, yielding a flexible and fault-tolerant system, featuring totally transparent parallelization from the users' point of view, both in a semantic and a syntactic sense.

PSM automatically creates and maintains an environment in which MRIP can be executed, as well as parallelizes and runs the simulation. Launching one simulation replication by activating a simulation program equipped with the necessary objects of Control and PSM creates a simulation engine. PSM provides dynamic binding between simulation engines and global control processes. Development of an efficient, portable and flexible Interprocess Communication (IPC) subsystem of PSM was regarded as the critical factor for achieving high efficiency in AKAROA. It is known that a careless implementation of IPC can even reduce speed of parallel processing, if high IPC overhead is generated. UNIX facilities for implementing IPC mechanisms include streams, pipes, socket-pairs, and various types of sockets (Quarterman, Silberschatz and Peterson 1985). AKAROA's IPC subsystem must support communicating processes located on different machines, and possibly belonging to different file systems. The IPC selected for AKAROA uses an extension of a (synchronous) Remote Procedure Call (RPC) mechanism decribed by Brian, Anderson and Lazowska (1990), and appears as an intermachine interprocess communication based on UNIX Internet domain datagram-type sockets that allow for fully file-less exchange operations.

4 PERFORMANCE OF MRIP

Dynamic properties of AKAROA, and the quality of SA-PTS estimators, were tested in a series of 1600 benchmark simulation experiments using P=1, 2, 4, and 6 processors. Initial studies of AKAROA's performance were done on a local computer network (a multiprocessor SUN Server with two SPARC CPUs, various SUN 4 and SUN SPARC workstations) based on Ethernet. Apart from the obvious differences in processing power between the workstations available for our investigations, none of the machines was dedicated solely to AKAROA's use.

In this situation, simulation experiments that we conducted for evaluating AKAROA on single processors (P=1) were done using the fastest machine available, during its low load periods, while all multimachine experiments involved a mix of the fast and lower rated workstations during normal working hours. Further, the priority of simulation processes engaged in parallel simulations was lower, to accommodate other users of the network, while the non-parallel simulations were run at the highest priority level. Thus, the results reported here are very conservative.

All results presented here were obtained from steady-state simulations of M/M/1/∞ queueing systems with traffic load ρ=90%. Each result is an average over 200 experiments. The measure estimated was the mean time spent in the system. The required level of precision of final estimates was $\leq$ 5%, at the 0.95 level of confidence. We tested also two different strategies for determining the distance between consecutive checkpoints during the simulation: one in which that distance was geometrically increasing and another one, in which it was kept constant. While former strategy is typically used in sequential steady-state simulations run on single processors, our results clearly show that the latter strategy (to keep checkpoints uniformly spaced) is much better in the MRIP scenario.

The results showing the real time speedup of simulations achieved with AKAROA, as a function of the number of workstations used, are depicted in Figure 1 and 2.

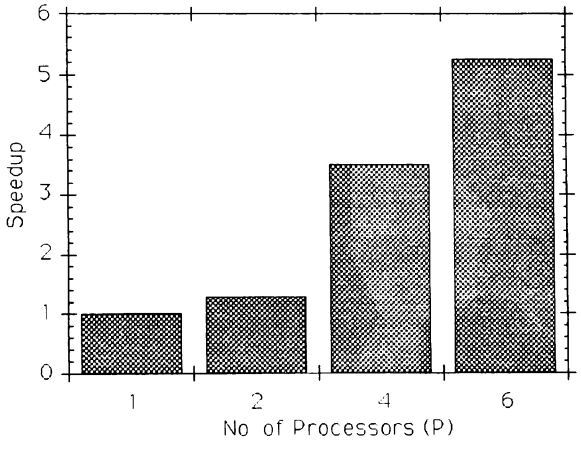

Figure 1: Real time speedup vs the number of processors employed. Geometrically spaced checkpoints.

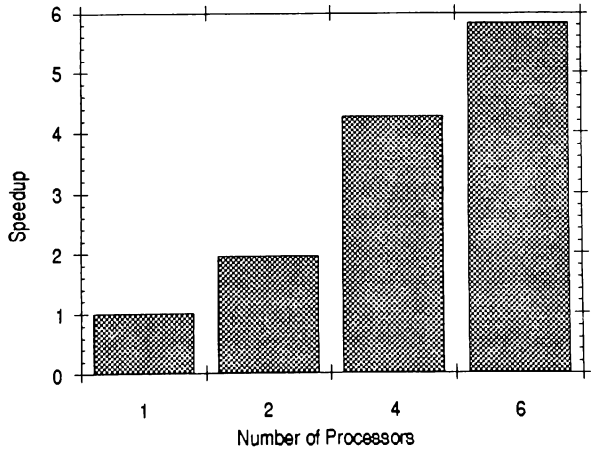

Figure 2: Real time speedup vs the number of processors employed. Uniformly spaced checkpoints.

Both figures show the nearly linear speedup offered by AKAROA in the case of geometrically spaced checkpoints, and even better speedup in the case of uniformly spaced checkpoints.

The CPU-times, normalized to the average time required for generating an observation, for different levels of parallelization, are compared in Figure 3 and 4. There are three results for each value of P: the average minimum run length of a simulation engine within 200 repeated MRIP simulations; the average number of observations per simulation engine produced during an experiment (averaged over 200 replications); and the average maximum run length of a simulation engine. Comparing the maximum replication lengths as a function of P, one can see that the reduction in CPU time with P workstations is greater than 1/P, suggesting super-linear speedup (!). This may be due to the fact that AKAROA uses CPU time more efficiently, and n observations generated by P workstations in parallel case (P>1) have higher entropy than if they were collected from a single replication. The results also show that using uniformly spaced checkpoints we can achieve substantially shorten simulation runs.

Finally, Figure 5 and 6 show the average numbers of messages (datagrams) exchanged in AKAROA as a function of P. One can see, the communication overhead grows slower than linearly with the number of communicating processors in the case of geometrically spaced checkpoints, and becomes practically constant for uniformly spaced checkpoints, showing a clear advantage of MRIP over traditional SRIP scenario, and a good efficiency of our IPC subsystem implemented in AKAROA.

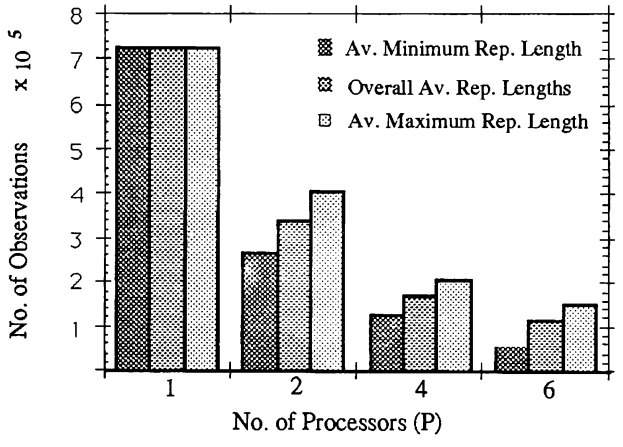

Figure 3: Speedup measured by reduction in CPU time vs the number of processors employed. Geometrically spaced checkpoints.

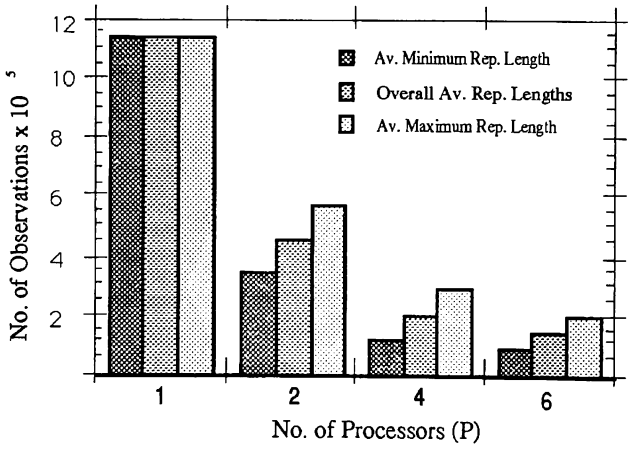

Figure 4: Speedup measured by reduction in CPU time vs the number of processors employed. Uniformly spaced checkpoints.

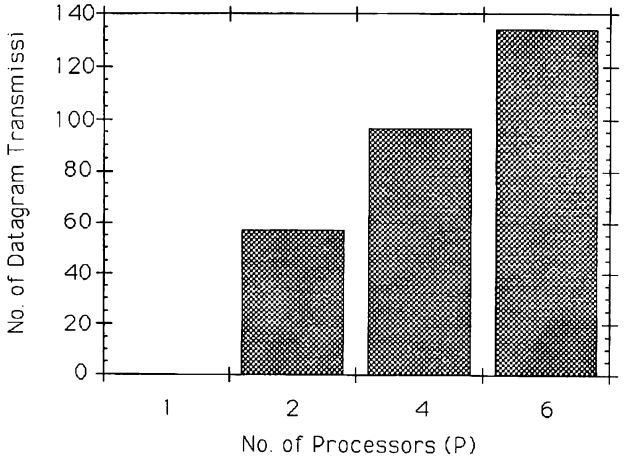

Figure 5: Average number of datagrams exchanged vs the number of processors employed. Geometrically spaced checkpoints.

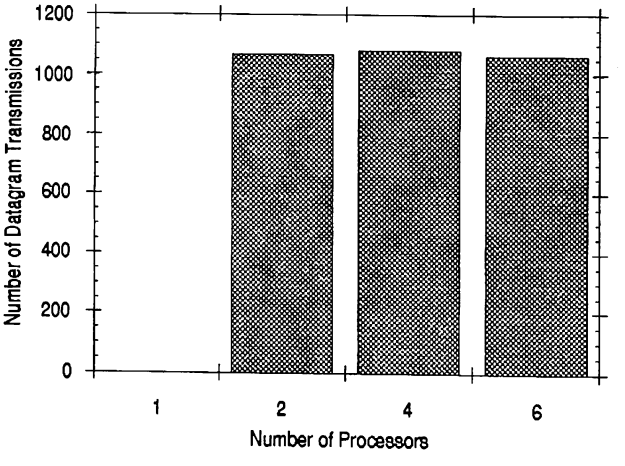

Figure 6: Average number of datagrams exchanged vs the number of processors employed. Uniformly spaced checkpoints.

4.1. Limitations of MRIP.

It is possible that when MRIP is applied in a heterogenous network, with one processor much faster than others, slower processors may not be able to contribute in parallel production of data since none of them would reach its first checkpoint when the fastest processor stops the whole simulation by generating the required number of observations.

On the other hand, the best speedup should be observed in homogeneous networks, with all processors (simulation engines) operating at the same speed. It is possible that in such situation all P processors would equally contribute in MRIP simulation, submitting, on average, the same number of observations to global analysers. Since the number of observations needed to obtain the required precision of results is fixed, at some stage each processor will be able to reach only the first checkpoint, and the simulation will be stopped. Let P_{max} be the minimum number of processors when this happens. In such a situation, adding more processors would not increase the speedup that has already reached its limit value equal P_{max}. The only effect of having more data (generated by $P>P_{max}$ processors) would be better final precision of results.

If n_1 is the mean number of observations, per simulation engine, needed to reach the first checkpoint, and N_{max} is the total number of observations needed for stopping the simulation with the required precision of results, then $P_{max}=\lceil N_{max}/n_1 \rceil$. Basing on our experience it means that simulating such dynamic queueuing system as M/M/1, loaded in 90 %, for typical values n_1=1 000, and N_{max}=1 000 000, we get P_{max}=1000. When simulating data communication networks such as DQDB (a standard fiber optic metropolitan area network operating at 150 Mbps) loaded in 50 %, for typical values n_1=500 and N_{max}=50 000, one could achieve the maximum speedup P_{max}=100. These observations are yet to be confirmed by experimental studies.

Let us note that if a distributed simulation originally based on SRIP is speeded up by factor S , then applying additionally MRIP, i.e. parallelizing P times a simulation model already distributed according to SRIP.F or SRIP.M, will additionally increase the speedup P times, i.e. the final speedup would be PS, as long as $P \le P_{max}$.

5 CONCLUSIONS

We have discussed the main features of a "forgotten" scenario for distributed quantitative stochastic simulation, named Multiple Replications in Parallel, and compared it with traditional Single Replication in Parallel. The most important features of MRIP are its universality, as it can be applied without exemption to any simulation model, and the high level of speedup it offers.

We have proposed a new technique of output data analysis (SA-PTS) and implemented it in AKAROA, our simulation package for automatic generation and control of processes for parallel stochastic simulation. The selection of SA/HW for sequential analysis of simulation output data was motivated by our intention of full automation of distributed quantitative steady-state simulation. This eliminated for example the method of regenerative cycles and independent replications, both adopted in EcliPse by Rego and Sunderam (1991, 1992) and Sunderam and Rego (1992), that require some decisions to be undertaken by simulators before a simulation, and, to make these decisions properly, one should know well the dynamics of the simulated system. Further design issues of AKAROA, as well performance evaluation of SA-PTS are under our current considerations.

ACKNOWLEDGEMENTS

The authors thank Dr Gill Bryant for her help in the final preparation of the paper. This research was partially supported by the Research Laboratories of Australian and Overseas Telecommunic-ations Co. in Melbourne, Australia. The permission of the Managing Director of AOTC to publish this paper is hereby acknowledged.

REFERENCES

Biles, W.E, C.M. Daniels, and T.J. O'Donnell. 1985."Statistical Considerations in Simulation on a Network of Multicomputers. Proc. 1985 Winter Simulation Conf., IEEE Press: 388-393.

Brian N.B., T.E.Anderson, and E.D.Lazowska. 1990. "Lightweight Remote Procedure Call". *ACM Transactions on Comput. Systems* 8: 37-55

Briner, J. 1988. "A Framework for Analysing Parallel Discrete Event Simulation". Proc. Int. Conf. Manag. and Perf. Evaluation of Computer Systems, CMG'88: 180-185.

Burk, W.H. 1990. "Limitations to Parallel Processing". Proc. 9th Int. Phoenix Conf. Computers and Communications, IEEE Press: 86-93

Comfort, J., and A.Miller. 1981. "Considerations in the Design of a Multiprocessor-Based Simulation Computer". In *Modelling and Simulation on Microcomputers*, ed. L. Leventhal, So. of Computer Simulation, laJolla, 1981.

Fujimoto, R. 1990. "Parallel Discrete Event Simulation". *Communications of the ACM*, 33: 30-60

Glynn, P.W., and P.Heidelberger. 1991. "Analysis of Parallel Replicated Simulations under a Completion Time Constraint". *ACM Transactions on Modeling and Computer Simulation*, 1: 3-23.

Heidelberger, P., and P.D. Welch. 1981. "A Spectral Method for Confidence Interval Generation and Run Length Control in Simulations". *Communications of the ACM* 25: 233-245.

Heidelbereger, P. 1986."Statistical Analysis of Parallel Simulations". Proc. 1986 Winter Simulation Conf.,

IEEE Press: 290-295
Heidelbereger, P. 1988. "Discrete Event Simulations and Parallel Processing: Statistical Properties". *SIAM Journal of Statistical Computing* 9: 1114-1132
Lubachevsky, B.D. 1989. "Efficient Distributed Event-driven Simulations of Multiple-Loop Networks". *Communications of the ACM* 32: 111-123
Pawlikowski, K. 1990. "Steady-State Simulation of Queueing Processes: a Survey of Problems and Solutions". *ACM Computing Surveys* 22: 123-170.
Pawlikowski , K., and V.Yau. 1991. "Independent Replications versus Spectral Analysis of Output Data in Steady-State Simulation of High Speed Data Networks". Proc. 6th Australian Teletraffic Research Seminar, University of Wollongong Press, Australia: 322-330.
Pawlikowski, K., and V.Yau. 1992. "An Automatic Partitioning, Runtime Control and Output Analysis Methodology for Massively Parallel Simulations". Proc. European Simulation Symposium, ESS'92, So. of Computer Simulation: 135-139
Quarterman, J.S., A.Silberschatz, and J.L.Peterson. 1985. "4.2BSD and 4.3BSD as Examples of the UNIX System", *ACM Computing Surveys* 17: 379-418
Rego, V.J., and V.S.Sunderam. 1991. "Concurrent Stochastic Simulation: Experiments with Eclipse". Proc. Int. Conf. Perferformance of Distributed Systems and Integrated Communication Networks: 253-271.
Rego, V.J., and V.S.Sunderam. 1992. "Experiments in Concurrent Stochastic Simulation: the Eclipse Paradigm". *Journal of Parallel and Distributed Computing* 14: 66-84
Stacey, C., K. Pawlikowski, and D. McNickle. 1993. "Detection and Significance of Initial Transient Period in Quantitative Steady-State Simulation". Proc. 7th Australian Teletraffic Research Seminar, Melbourne, RMIT Press: 193-202
Sunderam, V.S., and V.J.Rego. 1991. "EcliPse: a System for High Performance Concurrent Simulation". *Software-Practise and Experience* 21: 1189-1219
Wagner, D.B., and E.Lazowska. 1998. "Parallel Simulation of Queueing Networks: Limitations and Potentials". *Performance Evaluation Review* 17: 146-155.
Yau, V., and K.Pawlikowski. 1993. "AKAROA: a Package for Automatic Generation and Process Control of Parallel Stochastic Simulation". In Proc. 16th Australian Computer Science Conf., ed. G.Gopal et al., *Australian Computer Science Communications*: 15: 71-82

AUTHOR BIOGRAPHIES

KRZYSZTOF PAWLIKOWSKI is a Senior Lecturer in the Department of Computer Science at University of Canterbury, New Zealand. His research interests include quantitative stochastic simulation, and performance modelling and evaluation of telecommunication networks. He received a PhD in Computer Engineering from the Technical University of Gdansk, Poland, in 1975. Senior member of IEEE

VICTOR W. C. YAU is currently pursuing a PhD studies in the Department of Computer Science, University of Canterbury, New Zealand. His interests include simulation, computer architecture, telecommunication networks, and distributed systems. A student member of IEEE.

DON MCNICKLE is a Senior Lecturer in Management Science on the Department of Management, University of Canterbury, New Zealand. His research interests include queuing theory and statistical aspects of simulation. He received a PhD in Mathematics from the University of Auckland, New Zealand, in 1974. Member of ORSA.