AKAROA2: A CONTROLLER OF DISCRETE-EVENT SIMULATION WHICH EXPLOITS THE DISTRIBUTED COMPUTING RESOURCES OF NETWORKS

Don McNickle
Management Department
University of Canterbury
Private Bag 4800, Christchurch
New Zealand
Don.McNickle@canterbury.ac.nz

Krzysztof Pawlikowski and Greg Ewing
Computer Science and Software Engineering
University of Canterbury
Private Bag 4800, Christchurch
New Zealand
Krys.Pawlikowski@canterbury.ac.nz

KEYWORDS
Discrete-event simulation, sequential simulation, statistical analysis, multiple replications in parallel.

ABSTRACT

This paper describes and summarises our research on enhancing the methodology of automated discrete-event simulation and its implementation in Akaroa2, a controller of such simulation studies. Akaroa2 addresses two major practical issues in the application of stochastic simulation in performance evaluation studies of complex dynamic systems: (i) accuracy of the final results; and (ii) the length of time required to achieve these results. (i) is addressed by running simulations sequentially, with on-line analysis of statistical errors until these reach an acceptably low level. For (ii), Akaroa2 launches multiple copies of a simulation program on networked processors, applying the Multiple Replications in Parallel (MRIP) scenario. In MRIP the processors run independent replications, generating statistically equivalent streams of simulation output data. These data are fed to a global data analyser responsible for analysis of the results and for stopping the simulation. We outline main design issues of Akaroa2, and detail some of the improvements and extensions to this tool over the last 10 years.

INTRODUCTION

Quantitative discrete-event stochastic simulation is a useful tool for studying performance of stochastic dynamic systems, but it can consume much time and computing resources. Even with today's high speed processors, it is common for simulation jobs to take hours or days to complete. Even then, the results may not satisfy the decision-maker's objectives unless a proper experimental framework is set up which guarantees the results with an appropriate degree of accuracy.

Processor speeds are increasing as technology improves, but there are limits to the speed that can be achieved with a single, serial processor. To overcome these limits, parallel or distributed computation is needed. Not only does this speed up the simulation process, in the best case proportionally to the number of processors used, but the reliability of the program can be improved by placing less reliance on individual processors.

One approach to parallel simulation is to divide up the simulation model and simulate parts of it on different processors. However, depending on the nature of the model it can be very difficult to find a way of dividing it up, and if the model does not divide up readily, the overhead of communication between dependent parts of a given simulation model can even make simulation longer. Akaroa2 takes a different approach, applying *multiple replications in parallel* or MRIP. Instead of dividing up the simulation model, multiple independent instances of a given model are executed simultaneously on different processors. These instances continuously send output data (observations) back to a central controller/analyzer, measuring the performance parameters of interest. The central analyzer calculates overall estimates from these observations, e.g. the mean values of the parameters of interest. When it judges that it has enough observations to form all estimates with the required accuracy, it halts the simulation.

Since the simulation replications run independently, n copies of the simulation running on n processors will on average produce observations at n times the rate of a single copy. Therefore final results with an acceptably small statistical error can be produced much faster than from a single instance of the simulation. The total speedup depends on the type of simulation (terminating or steady-state), the type of estimators, and on the method of estimation. A Truncated Amdahl's Law which captures this is formulated in Pawlikowski and McNickle (2001).

MRIP also provides some degree of fault tolerance. It does not matter which instance of the simulation the estimates come from, so if one processor fails, the program it was running can be restarted and the simulation continued without penalty. Alternatively, the simulation will simply continue with one less processor and take proportionately longer to complete. If a simulation is taking an unacceptably long time to complete, additional processors can be called in at any time (note the "Add Engines" button in Figure 3.)

PROGRAM ARCHITECTURE

The main components of Akaroa2 are the *akmaster*, the *akslaves*, *akrun* and the *simulation engines*. The relationships between these components are shown in Figure 1.

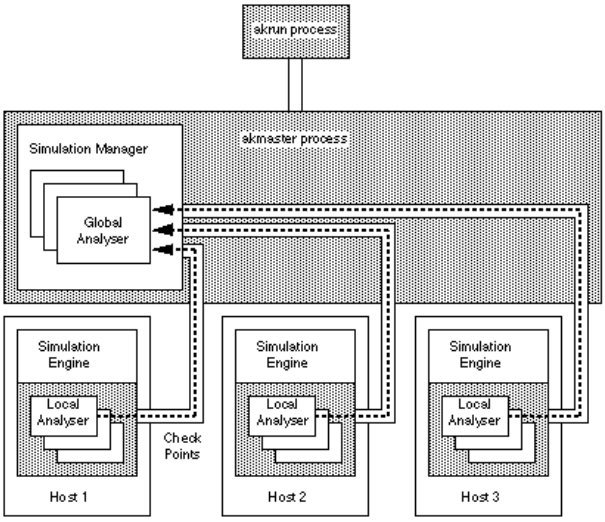

Figure 1. The basic structure of Akaroa2

The *akmaster* process coordinates the activity of all other processes initiated by Akaroa2. It launches new simulations, maintains state information about running simulations, performs global analysis of the data produced by simulation engines, and makes simulation stopping decisions.

Akslave processes (not shown) run on the hosts which are to run the simulation engines. The only function of the akslave is to launch simulation engine(s) on its host as directed by the akmaster. The akslave processes have been introduced because other methods of launching remote processes under UNIX (for example, by using *rsh*) tend to be slow and unreliable.

The *akrun* program is used to initiate a simulation. It first contacts the akmaster process, obtaining its host name and port number from a file left by the akmaster in the user's home directory. For each simulation engine requested, the akmaster chooses a host from among those hosts on the LAN which are running akslave processes. It instructs the akslave on that host to launch an instance of the user's simulation program, passing on any specified arguments. The first time the simulation program calls one of the Akaroa2 library routines, the simulation engine opens a connection to the akmaster process and identifies the simulation to which it belongs, so that the akmaster can associate the connection with the appropriate simulation data structure.

Each engine performs sequential analysis of its own data to form a local estimate of each performance measure. At more or less regularly determined *checkpoints*, the engine sends its local estimates to the akmaster process, where the local estimates of each performance measure from all engines are combined to give a set of *global estimates*.

Whenever a new global estimate is calculated, the relative half-width of its confidence interval at the requested confidence level is computed, and compared with the requested precision. When the precision of all analysed performance measures becomes satisfactory, the akmaster terminates all the simulation engines, and sends the final global estimates to the akrun process, which in turn reports them to the user. Several different simulation experiments may be run simultaneously, using separate *akrun* processes. They may launch instances on the same or different hosts.

In Akaroa2, all interprocess communication is via TCP/IP stream connections, which provide reliable, sequenced, non-duplicated delivery of messages.

An earlier version, Akaroa, used UDP/IP datagrams to communicate between processes. Since the UDP protocol does not guarantee reliable packet delivery, Akaroa spent a great deal of effort attempting to deal with issues of packet loss and duplication. The system was unreliable and difficult to manage. If a process failed to respond within an arbitrary timeout, it was hard to tell whether it had died or was simply taking longer than usual to respond.

The program is implemented in the GNU dialect of C++, and has been tested at the University of Canterbury under the SunOS 4, Solaris 2 and Linux operating systems. Further operating details can be found in the manual, downloadable from the Akaroa2 website at http://www.akaroa2.canterbury.ac.nz/.

The User Interface

Originally, Akaroa2 could only be run by running akrun directly with command-line arguments. Now a graphical user interface *akgui* is provided. The main screen for the akmaster process is shown in Figure 2. This shows a single simulation experiment, being run on 5 engines.

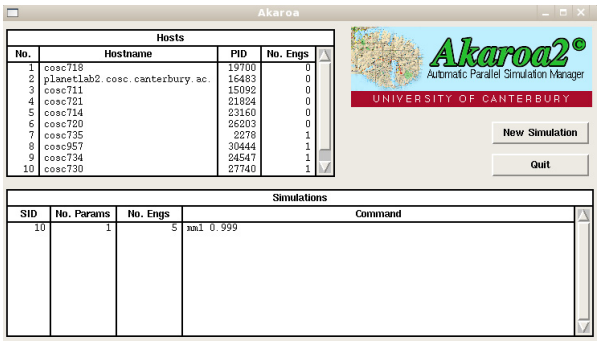

Figure 2. The Main Screen for the Akmaster Process

The progress of each simulation being executed is shown in a progress window (Figure 3). The bar graph shows that the relative precision of (in this case a single) performance measure is currently about 0.2, converging towards the requested level (in this case a 95%

confidence level being estimated to a relative precision of 0.05.)

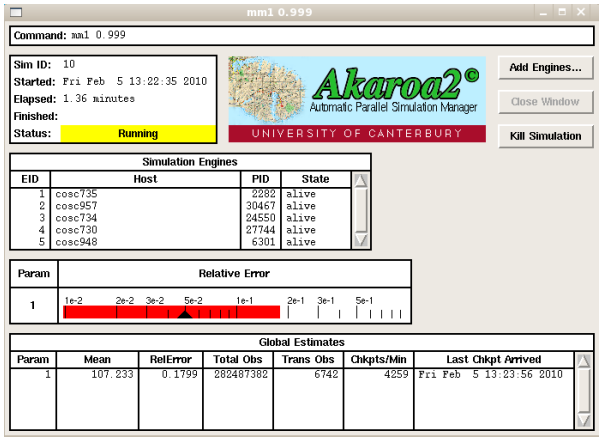

Figure 3. The Akgui Progress Window

IMPROVEMENTS TO INTERFACES AND OPERABILITY

Interfaces with Other Simulation Packages

A popular application of Akaroa is to use it as a controller controlling execution of simulations based on models built with help of other simulation packages. Currently interfaces allow linking Akaroa2 with Ptolemy Classic, the Network Simulator NS2, and OMNeT++. Details of these interfaces can be found from the Akaroa2 website.

Simulation on wide area networks

While Akaroa2 is normally used on a local network, there is nothing in principle to prevent a simulation from being distributed over any set of hosts that can reach each other via the Internet. The communication between processes in an Akaroa2 simulation only requires a very low bandwidth, and is mostly one-directional, so network delays have little influence. In theory, therefore, there should be little difference in performance between using a local, or a wide area network. To test this, Akaroa2 simulations have been run on geographically separated PlanetLab research network nodes (Yasmeen, Ewing, Pawlikowski, and Yamada, 2009). Because of the use of TCP/IP as the communication protocol implementation of this was straightforward. Apart from a slight increase in the time taken to start the simulation up, there was no discernable difference in simulation speed, as predicted.

IMPROVEMENTS TO SIMULATION AND STATISTICAL ASPECTS

The basic approaches to statistical problems of estimation and control, which were adopted in Akaroa2, are described in Pawlikowski (1990). Further details are given in Ewing, Pawlikowski and McNickle (1999). However there have been a number of substantial modifications since these papers.

The Internal Simulation Routines

The simulation library in Akaro2 is now based on a process-interaction approach. This replaces the previous event-scheduling approach, although this remains available. A Process Manager creates processes for each class of entities, describing the life cycle through which these entities go. The class Resource is used to represent finite classes of (usually permanent) entities, to model competition for access to the resource. The class Queue implements a queue of entities of some type, which entities join and leave, with priorities if desired. The usual range of random-variate generators are available. More complex simulations can be written in any language as long as the program is capable of calling routines written in C. Akaroa2 simply picks up, analyses and controls the output observations from this program.

Random Number Generation

In MRIP, each simulation engine must use pseudo-random numbers (PRNs) independent from those used by other engines. Consequently, in Akaroa2, random number generation is not left to the user's simulation program. Instead, the akmaster process is given full control over PRNs used by different simulation engines. Currently Akaroa2 uses a Combined Multiple Recursive PRN Generator with a period of approximately 2^{191}. This sequence is divided into blocks of 2^{128}, and different blocks of PRNs are assigned to different simulation engines. Thus, one could concurrently use up to 2^{67} replications, providing that none of them requires more than 2^{128} PRNs. If fewer replications are used and a simulation engine requires more PRNs, it would receive next block of PRNs. The particular generator used by Akaroa2 is known as MRG32k3a (see, for example L'Ecuyer (1999)). Since it would be very inefficient for a simulation engine to have to communicate with the akmaster process every time it wanted a PRN, each engine generates its assigned block of PRNs by itself, initialising its own copy of the generator from the starting values assigned by the akmaster.

Confidence Interval Estimation

Akaroa2 provides two methods of sequential analysis of output data from steady-state simulation: an automated non-overlapping Batch Means algorithm, and a Spectral Analysis method based on Heidelberger and Welch's (1981, 1983) results. Its implementation in MRIP is described in Ewing, McNickle and Pawlikowski (2002). Some further modifications which improve its performance for sequential analysis are described in McNickle, Ewing and Pawlikowski (2004). Figures 4

and 5 compare the results for simulating an M/M/1 queue using our Batch Means method (Figure 4) and the modified version of Spectral Analysis (Figure 5). What is plotted here is the coverage – that is the actual estimated size of (in this case supposedly 95%) confidence intervals for the mean waiting time, produced by calculating from typically 10,000 separate experiments, the fraction of confidence intervals that actually contain the true mean waiting time. The fall-off in the batch means coverage with load can be entirely explained by correlation between the batch means, (the dashed lines in Figure 4) show the theoretical loss of coverage) which remains a risk with most batch-mean methods. In contrast the coverage from modified Spectral Analysis is almost always at the level specified. Thus our research leads us to strongly favour the Spectral Analysis approach, see also Pawlikowski, McNickle and Ewing, (1998).

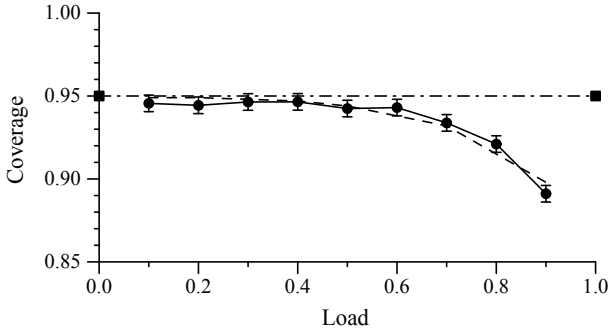

Figure 4. Coverage with Batch Means

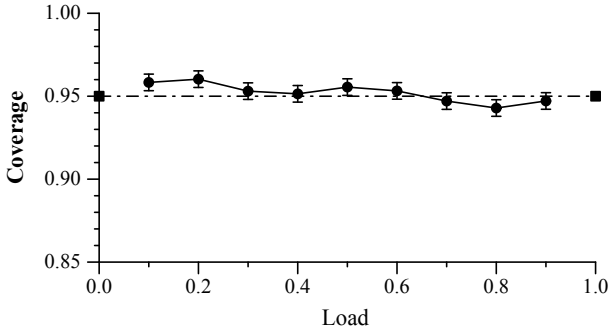

Figure 5. Coverage with Modified Spectral Analysis

An additional attraction of Spectral Analysis is that the same algorithm applies exactly to batched data. This batching of data reduces storage and communications costs.

Transient Period Detection

When Akaroa2 controls a steady-state simulation, an automated method is used to determine the length of the initial transient period. It begins with a heuristic proposed by Gafarian, Anker and Morisaku (1978) to decide when to start testing for stationarity. Its use in a sequential context is described in detail in Pawlikowski. In this heuristic, the length of initial transient period is first taken to be over when the sequence has crossed its running mean 25 times. Then a sequential version of Schruben's test (Schruben, Singh and Tierney, 1983) is used to test for stationarity. If the null hypothesis of stationarity is rejected, the length of the potential transient period is doubled and the test repeated (Pawlikowski). Comparisons with a limited range of other transient deletion methods can be found in Pawlikowski, Stacey and McNickle (1993) which show that this method, although simple, does appear to work well, at least for basic queueing models.

However detecting the length of the transient period remains an uncertain area of discrete-event simulation theory. A large number of methods for selecting the number of observations to delete, and for testing if the system is adequately close to "steady state", have been proposed. Hoad, Robinson and Davies (2008) list 42 methods. Some of these proposals appear to have had limited testing, so their validity remains in question. While our current method appears to be adequate for most simple models, we are considering a sequential version of MSER-5, given its good performance reported in Hoad, Robinson and Davies.

Estimating Quantiles

Mean values provide very limited information about the analysed processes. Much more meaningful insight is provided by quantiles, especially if several quantiles can be estimated simultaneously. For example, 90th or 95th percentiles are often specified by decision makers as the criteria of quality when considering delays or overflow probabilities in manufacturing, customer service, emergency response or telecommunication systems. Estimation of quantiles is yet another order of magnitude harder than estimation of means or variances, as now large amounts of data need to be stored and efficiently sorted. As well as the usual problems due to serial correlation of output, the estimates of multiple quantiles estimated from a single run will also be correlated. Thus, the use of independent replications on this problem is especially attractive.

Lee, Pawlikowski and McNickle (2000) reports on using the existing methods in Akaroa2 to directly estimate quantiles, with reasonable success. However a more advanced method is described below.

Avoiding Premature Stopping

A chronic problem of sequential simulation is that some of the simulation experiments may stop with an insufficient number of observations because, by chance, the required accuracy is apparently temporarily attained. As a result the actual precision of the results obtained is less than specified.

Figure 6 illustrates the problem. It plots the estimated relative precision of a simulation estimating the mean waiting time in a queue, against the (geometric)

checkpoint number. The horizontal lines show stopping criteria of relative precisions of 0.1 and 0.05. Using a relative precision of 0.1 would result in the simulation very nearly stopping at about the 30th checkpoint, whereas data from at least 200 checkpoints are needed. It is also worth noting, as Figure 6 suggests, that this problem is reduced if very high accuracy (low relative precision) is specified – note how the eventual convergence to a relative precision of 0.05 is much less erratic than that to 0.1. High accuracy implies a large amount of data of course, so MRIP has a valuable role in ameliorating this problem by providing large amounts of data, and hence accurate and reliable results, in reasonable elapsed time.

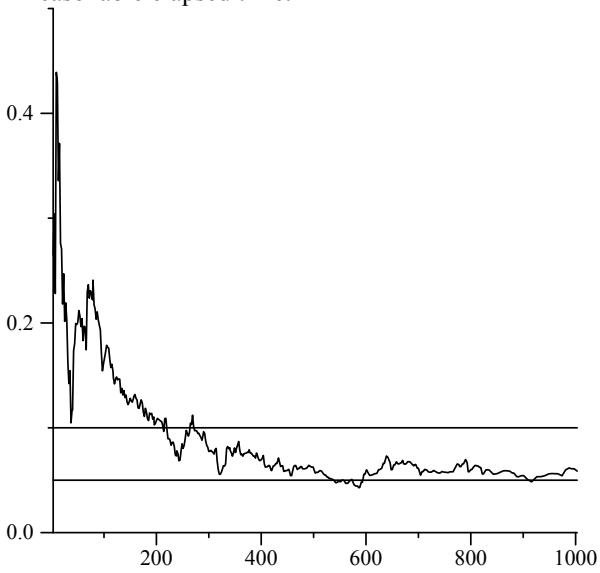

Figure 6. The Risk of Premature Stopping in Sequential Simulation

A recent study (McNickle, Pawlikowski and Ewing, 2010) has suggested that at the run lengths required by common levels of accuracy, the bias in the mean may be negligible. However the coverage can be seriously affected due to prematurely stopped runs. Using a reliable transient deletion technique, and less certainly, initial loading, turns out to help here. The improvements to the Spectral Analysis technique in McNickle, Pawlikowski and Ewing (2004) also help with this problem. In addition a range of simple rules of thumb are described in Lee, Pawlikowski and McNickle (1999) that make use of the multiple replications provided by Akaroa2. The best of these (in its simplest form: take the values from the longest of a number of completely independent runs) has been implemented in the program.

IMPROVEMENTS CURRENTLY IN PROGRESS

Quantile Estimation

In Eickhoff, Pawlikowski and McNickle (2007a) two methods for estimation of multiple quantiles using parallel replications on networks of computers are described (see also Eickhoff, McNickle and Pawlikowski, 2006.) These can be used to simultaneously estimate multiple quantiles, with the set of quantiles to be estimated selected automatically if required. They are in the process of being implemented in a newer version of our simulation controller.

Transient Detection

Almost all methods for detecting the initial transient have been tested on mean values only. Once we move to other measures, demonstrating mean-stationarity may not be enough to determine an appropriate transient period to delete, and methods that demonstrate stationarity in distribution are required. A distribution-based method using the techniques reported in Eickhoff, Pawlikowski and McNickle (2007a), which specifically makes use of parallel replications, is under development.

Estimating Variances

Current analysis of output data from discrete event simulation focuses almost exclusively on the estimation of mean values, largely for reasons of speed, ease of analysis and minimal data storage requirements. However there are a number of applications for which we require the variance – for example jitter in video streams, safety stock in inventory problems, etc. In Schmidt, Pawlikowski and McNickle (2009) three methods of point and interval estimation of the steady-state variance are considered. One, based on splitting of sums of squares, appears to be superior. Estimating variances involves considerably more observations than estimating means. Thus, selecting estimators with good performance characteristics, and using MRIP, is even more important.

CONCLUSIONS

Akaroa2 is downloadable for teaching, and for non-profit research (by universities only) from http://www.akaroa2.canterbury.ac.nz/. It has been downloaded more than 1800 times since the counter was introduced in 2001. Without requiring the use of any parallel programming techniques, it automatically distributes simulation models over an arbitrary number of computers linked by a network, and controls the simulation run length so as to produce final results having a specified precision, both in the case of terminating and steady-state simulation.

REFERENCES

Eickhoff, M., McNickle, D. and Pawlikowski, K. 2006. "Analysis of the Time Evolution of Quantiles in Simulation", *Int. J. Simulation* 7 (6) (2006), 44-55.

Eickhoff, M., Pawlikowski, K. and McNickle, D. 2007a. "Detecting the Duration of Initial Transient in Steady State Simulation of Arbitrary Performance Measures", in

Proceedings ACM ValueTools07 (23-25 October 2007), Nantes, France.

Eickhoff, M., Pawlikowski, K. and McNickle, D. 2007b. "Using Parallel Replications for Sequential Estimation of Multiple Steady State Quantiles", in *Proceedings ACM ValueTools07*, (23-25 October 2007), Nantes, France.

Ewing, G., McNickle, D. and Pawlikowski, K. 2002. "Spectral Analysis for Confidence Interval Estimation under Multiple Replications in Parallel", in *Proc. 14th European Simulation Symposium*, Dresden (October 2002), 52-61.

Ewing, G., Pawlikowski, K. and McNickle, D. 1999. Akaroa2: "Exploiting Network Computing by Distributed Stochastic Simulation" in *Proc. 13th European Simulation Multiconference,* Warsaw, Poland (June 1999), SCSC, 175-81.

Gafarian, A. V., Ancker, C. J. and Morisaku, T. 1978. "Evaluation of Commonly Used Rules for Detecting "Steady State" in Computer Simulation", *Naval Research Logistics Quarterly*, 78 (1978) 511-529.

Heidelberger, P. and Welch, P. D. 1981. "A Spectral Method for Confidence Interval Generation and Run Length Control in Simulations", *Communications of the ACM*, 24(4) (April 1981), 233-245.

Heidelberger, P. and Welch, P. D. 1983 "Simulation Run Length Control in the Presence of an Initial Transient", *Operations Research*, 31 (1983) 1109-1144.

Hoad, K., Robinson, S. and Davies, R. 2008 "Automating Warm-up Length Estimation", in *Proc. 2008 Winter Simulation Conference*, S.J. Mason et al. eds., (2008) 532-540.

L'Ecuyer, P. 1999. "Good Parameters and Implementations for Combined Multiple Recursive Random Number Generators", *Operations Research*, 47, 1, Jan-Feb 1999, 159-164.

Lee, J.-S. R., Pawlikowski, K. and McNickle, D. 1999 "Sequential Steady-State Simulation: Rules of Thumb for Improving the Accuracy of the Final Results", in *Proc. ESS99* (Erlangen, Germany, Oct 26-28 1999), 618-622.

Lee, J.-S. R., Pawlikowski K. and McNickle, D. 2000 "Initial Transient Period Detection for Steady-State Quantile Estimation", in *Proc. Summer Computer Simulation Conference* SCSC'2000, Vancouver, Canada, International Society for Computer Simulation, San Diego, July 16-20, 2000, Paper #S213, 1-6

McNickle, D., Pawlikowski, K. and Ewing, G. 2004 "Refining Spectral Analysis for Confidence Interval Estimation in Sequential Simulation" in *Proceedings of the ESS2004*, Budapest, Hungary Oct 2004, 99-103.

McNickle, D., Ewing, G. and Pawlikowski, K. 2010. "Some Effects of Transient Deletion on Sequential Steady-State Simulation", to appear in *Simulation Modelling Practice and Theory*, paper SIMPAT864.

Pawlikowski K., Stacey, C. and McNickle, D. 1993. "Detection and Significance of the Initial Transient Period in Quantitative Steady-State Simulation", in *Proc. Eighth Australian Teletraffic Research Seminar*, RMIT Melbourne, (6-8 December 1993) 193-202.

Pawlikowski, K., McNickle, D. and Ewing, G. 1998. "Coverage of Confidence Intervals from Sequential Steady-State Simulation", *Simulation Practice and Theory*, 6 (1998), 255-267.

Pawlikowski. K. And McNickle, D. 2001. "Speeding up Stochastic Discrete-Event Simulation", in *Proc. European Simulation Symposium, ESS'01*, Marseille, France, 18-20 October, ISCS Press, 132-138.

Pawlikowski, K., Schoo, M. and McNickle, D. 2006. "Modern Generators of Multiple Streams of Pseudo-Random Numbers", in *Proc. Int. Mediterranean Modelling Multiconference* (ESM06), Barcelona, 553-559.

Pawlikowski, K. 1990. "Steady State Simulation of Queueing Processes: a Survey of Problems and Solutions", *ACM Computing Surveys*, (22, June 1990) 123-170.

Schmidt, A., Pawlikowski, K. and McNickle, D. 2009. "Sequential Estimation of the Steady-State Variance in Discrete Event Simulation", in *Proc. ECMS 2009*, 630-635.

Schruben, L., Singh, H. and Tierney, L. 1983. "Optimal Tests for Initialisation Bias in Simulation Output", *Operations Research*, 31(6) (1983) 1167-1178.

Yasmeen, F., Ewing, G., Pawlikowski, K. and Yamada, S. 2009 "Distributing Akaroa2 on PlanetLab". Proceedings of IEICE General Conference, Matsuyama City, Japan, March 17-20, 2009.

AUTHOR BIOGRAPHIES

DON MCNICKLE is an Associate Professor of Management Science in the Department of Management at the University of Canterbury. His research interests include queueing theory; networks of queues and statistical aspects of stochastic simulation. He is a full member of INFORMS.

GREG EWING is an Adjunct Research Associate in the Department of Computer Science and Software Engineering at Canterbury; where received a Ph.D. His research interests include simulation; distributed systems; programming languages, 3D graphics and graphical user interfaces. He has made contributions to the Python programming language.

KRZYSZTOF PAWLIKOWSKI is a Professor of Computer Science at the University of Canterbury. His research interests include quantitative stochastic simulation; and performance modelling of telecommunication networks. He received a PhD in Computer Engineering from the Technical University of Gdansk, Poland. He is a Senior Member of IEEE and a member of SCS and ACM. His web page is <http://www.cosc.canterbury. ac.nz/"'krys/>.

ACKNOWLEDGEMENT

This research was supported in part by grants from the College of Business and Economics, and the College of Engineering, University of Canterbury.