Proceedings of the 1995 Winter Simulation Conference
ed. C. Alexopoulos, K. Kang, W. R. Lilegdon, and D. Goldsman

SIMULATING POPULATION DEPENDENT PCS NETWORK MODELS USING TIME WARP

Christopher D. Carothers
Richard M. Fujimoto

College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332, U.S.A.

Yi-Bing Lin

Department of Computer Science and
Information Engineering
National Chiao Tung University
Hsinchu, Taiwan, R.O.C.

ABSTRACT

The demand for mobile communications over the past few years has grown at a rapid pace. This demand has led to intensive research and development efforts for complex PCS (personal communication service) networks. Capacity planning and performance modeling is necessary to maintain a high quality of service to the mobile subscriber while minimizing cost to the PCS provider. A question of pragmatic interest concerns the modeling of subscriber or *portable* movement in a PCS network. Typically, portable movement models are based on a probabilistic distribution function with fixed mean. These models may be an over-simplification of real portable movement patterns. For example, portable movements during rush-hour traffic may be slowed due to traffic jams. However, when the volume of vehicular traffic is less, portables are allowed to move more freely and with greater speed. To capture this type of portable movement behavior, we develop a population dependent PCS model where portable movement is based on the number of portables currently residing in that service area or *cell*. Because of the large amount of computation required to simulate PCS networks, we implement this model on a distributed Time Warp simulator, which has been shown to reduce the execution time of a single run from 20 hours down to 3.5 hours. Using this simulation model, we study the effect of different call workloads and population dependent portable movement patterns on PCS blocking statistics and present our preliminary results.

1 INTRODUCTION

A personal communication service (PCS) network as described by Cox (1995) provides wireless communication services for nomadic users. The service area of a PCS network is populated with a set of geographically distributed transmitters/receivers called *radio ports*. A set of radio channels are assigned to each radio port, and the users in the *coverage area* (or *cell* for the radio port) can send and receive phone calls by using these radio channels. When a user moves from one cell to another during a phone call a *hand-off* is said to occur. In this case the PCS network attempts to allocate a radio channel in the new cell to allow the phone call connection to continue. If all channels in the new cell are busy, then the phone call is forced to terminate. It is important to engineer the system so that the likelihood of forced termination is very low (e.g., less than 1%).

Most PCS models fall into two types of models: *portable-initiate* (Lin and Mak 1993 and Wong 1993) or *call-initiated* (Chuang 1993 and Tekinary and Jabbari 1992). The *portable-initiated* model is organized around two object types: *cell* and *portable*. The cell represents a cellular receiver/transmitter that has some fixed number of channels allocated to it. The portable represents a mobile phone unit that resides within the cell for a period of time and then moves to one of the four neighboring cells. As shown in Figure 1, when a new call arrives at a cell, the cell first determines the status of the destination portable. If the destination portable is busy with another call, this call is counted as a *busy line*. A *busy line* occurs when a portable is currently connected in a phone call and another phone call arrives for that portable. If the portable is not busy, the cell determines channel availability. If all channels are busy, the call is counted as a *call block*. If a channel is available, it is allocated for the destination portable's use and the call is allowed to connect. While a call is in progress, the portable tracks its location. When the portable determines it is moving out of the current cell's signal range, it drops the currently used channel, and requests a channel from the neighbor cell into which it is moving. If all channels from the neighboring destination cell are busy, this call counts as a *hand-off block*. If a channel is available, the call

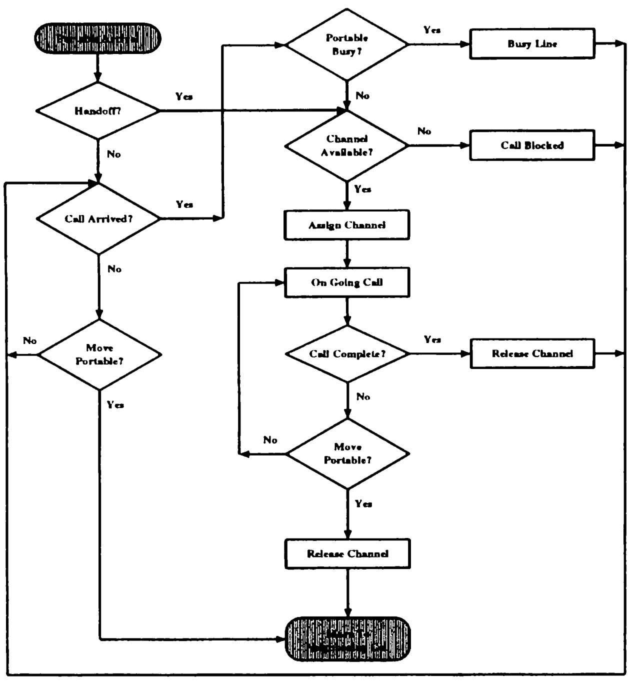

Figure 1: *Portable-initiated model*: flowchart for call processing within a single cell. A "Portable Arrival" denotes a portable entering a cell's area.

reconnects and continues without interruption. Now, where the *portable-initiated* model tracks the movement of portables even if they are not in conversations, the *call-initiated* model only simulates the behavior of a portable during phone conversations. Other functionalities in the *call-initiated* model are identical to the *portable-initiated* model.

Typically, portable movement is modeled using a probabilistic distribution with some fixed mean as done by Wong (1993). However, this model may oversimplify real portable movement patterns. Consider the following example: you are traveling to work in your car. Suddenly, the flow of traffic ahead of you comes to a halt and you find yourself in another morning rush-hour traffic jam. Realizing you could be late for work, you pick up your PCS hand-set to call your boss only to hear a recorded voice say "All channels are busy, please try again later". In this situation, a large number of portables have become struck in the same cell due to the traffic jam. Consequently, portable movement patterns have become dependent on the number of portables currently residing in the cell. To model this type of portable movement behavior, we use the *portable-initiated* PCS model since it keeps track of all portable movements.

To guarantee the simulation model has reached steady state and the network performance statistics are not biased it has been shown that the number of cells in the network should be large (greater than 256) and amount of simulated time should be greater than 5×10^4 simulated seconds (Carothers et al. 1994). To reduce the execution of these simulation models, we use a distributed simulator based on the Time Warp mechanism by Jefferson (1985).

The remainder of the study focuses on the effect of varying call workloads and mobility patterns on PCS network performance. In Section 2 we briefly describe our distributed Time Warp implementation. Section 3 describes the PCS network performance results and Section 4 presents conclusions of this study, and future work.

2 DISTRIBUTED TIME WARP

The distributed simulator consists of a collection of *logical processes* or LPs, each modeling a distinct component of the system being modeled. LPs communicate by exchanging timestamped event messages. Like most existing distributed simulation protocols, we assume different LPs may not share state variables that are modified during the simulation. The synchronization mechanism must ensure that each LP processes events in timestamp order in order to prevent events in the simulated future from affecting those in the past. The synchronization issue has been widely studied (e.g., see Fujimoto 1990 and 1992, Misra 1986, Richter and Walrand 1989). The Time Warp mechanism uses a detection-and-recovery protocol to synchronize the computation. For a more detailed discussion of the Time Warp mechanism we refer the reader to Fujimoto (1990) and Jefferson (1985).

A version of Time Warp has been developed that executes on a collection of DEC 5000 workstations, Sun Sparc workstations, SGI Indy workstations or a mixture of these machines. All performance results presented here were performed on DEC machines interconnected by an Ethernet. The Time Warp system is written in C++. A principal objective of this implementation is to enable efficient simulation of thousands of "light weight" LPs (i.e., processes that contain a small amount of state and perform little computation in each event) in an object-oriented environment on networked, heterogeneous workstations. Here we will describe some of the user and kernel level features of the Time Warp system.

LPs are implemented as C++ objects, and are referred to as "entities." Each LP (entity) consists of a state vector that stores the LP's private data, and a set of methods that define the allowable operations that can be performed on that data. Each method corresponds to a type of event.

It is anticipated that most simulations will contain far more entities than processors, so each processor will typically contain hundreds or thousands of entities (LPs). A priority queue data structure called the calendar queue by Brown (1988) is used in each processor to select the next entity to execute. The processor's scheduler always selects the entity containing the smallest timestamped event as the next one for execution. Each entity includes a linear list to hold the unprocessed events (method invocations) scheduled for that entity.

To avoid unnecessary system overheads from `malloc` system calls and memory fragmentation, the Time Warp kernel allocates a single contiguous block of memory from which events and other internal data structures are allocated.

Communications between processors is implemented using PVM 3.2 (Parallel Virtual Machine), a message passing system for heterogeneous collections of networked computers (Geist 1993). In addition to PVM's default message routing, PVM will route messages directly between application processes by setting up a TCP/IP connection between each enrolled application, thus bypassing the PVM daemons at both the sending and receiving host. In this mode, the sending process is not blocking, making PVM's direct routing mode an asynchronous communications protocol. When routing messages in direct mode, the total time to deliver a message is in the range of 1.4 to 2.0 milliseconds on a Sun Sparc-2. When executing our distributed Time Warp kernel, PVM is configured to directly route messages.

3 PERFORMANCE RESULTS

In this section we present performance data for the population dependent PCS simulation model. The definitions and notation that will be used throughout the remainder of this paper are as follows:

- The portable residual times in a coverage area are based on a probabilistic distribution with the mean $1/\eta$ multiplied by the number of portables that currently reside within that cell. The population dependent portable residual time distinguishes this PCS model from the PCS models discusses in our previous work.

- The call holding time is exponentially distributed with mean $1/\mu = 180$ seconds.

- The call interarrival times to an individual portable are exponentially distributed with mean $1/\lambda$.

- The aggregate call load to a cell is measured in Erlangs. An Erlang is $N \times \lambda/\mu$, where N is the average number of portables residing in that cell.

- p_f is the forced termination probability (the probability that a hand-off call is blocked).

- p_o is the probability that a new call is blocked.

- p_{nc} is the total call blocking probability. This probability includes both new call blocks and hand-off calls that are forced to terminate. An incomplete call is either blocked as a new call or it may make several successful hand-offs before it is forced terminated.

The speedup for this simulation model ranges between 2.8 and 7.8 using 8 DEC 5000 workstations (Carothers et al. 1994). Here, it was shown that speedup increases when (i) number of portables per cell increases, (ii) call interarrival time decreases, and (iii) mobility (η) decreases. For a detailed performance analysis of the *portable-initiated* and *call-initiated* PCS models executing on a Time Warp simulator, we refer the reader to Carothers et al. (1995).

To determine the effect of varying call workloads and population dependent mobility patterns on PCS blocking statistics, we conducted a series of experiments where the call interarrival times and mobility distribution and mobility interarrival times were varied. The number of channels per cell was fixed at 8. The initial number of portables assigned to each cell was also fixed at 25. Each experiment contained 1024 cells and ran for 2.5×10^5 simulated seconds to ensure that the output statistics where not biased. The results presented here represent over 330 runs of the distributed simulator.

The mobility distribution is varied over a range of 5 different probabilistic distributions: (i) exponential, (ii) Pareto with $\alpha = 1.6$, (iii) Pareto with $\alpha = 5.0$, (iv) Pareto with $\alpha = 100.0$ and (v) uniform. These distributions were chosen because they represent a wide range of statistical characteristics, such as the length of the tail and rate of decay. The exponential distribution is traditionally used to model portable movements and was originally suggested by Wong (1993).

To give the reader a feel for these distributions, Figures 2 and 3 plot the inverse cumulative distribution function against x values in the range of 0 to 1 for each distribution. The shape of the Pareto distribution, as shown in Figure 2, is controlled by the α parameter. When α is equal to 1.6 (Figure 2(a)), the variance of the Pareto distribution is infinite and has a very long tail. However, when α is equal to 5.0, the

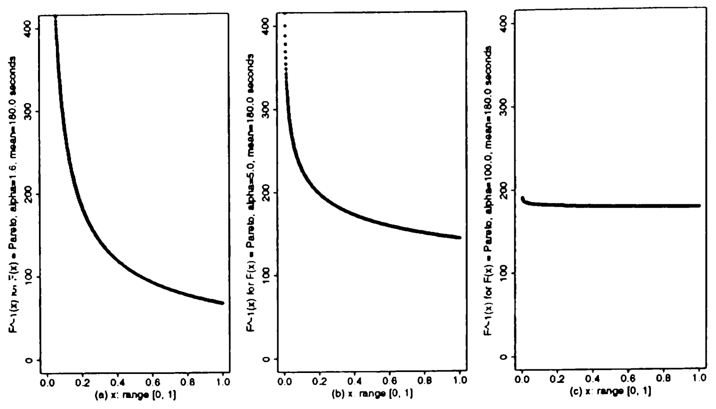

Figure 2: Inverse cumulative distribution function for Pareto with mean = 180.0 seconds for different values of α: (a) $\alpha = 1.6$, (b) $\alpha = 5.0$, (c) $\alpha = 100.0$.

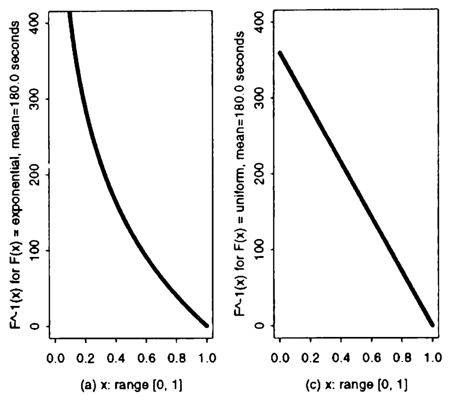

Figure 3: Inverse cumulative distribution functions as a function of x : $range[0, 1]$, mean = 180.0. (a) exponential, (b) uniform.

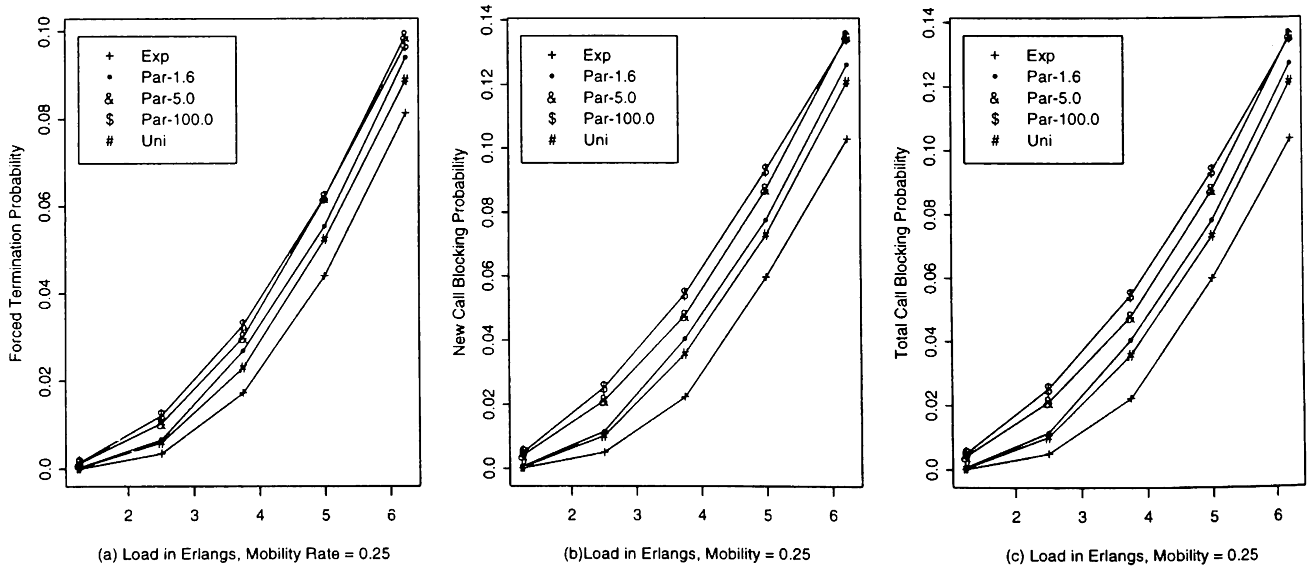

Figure 4: Effect of exponential, Pareto, and uniform mobility distributions on blocking probabilities as a function of λ in Erlangs. $\eta = 0.25\mu$. (a) p_f, forced termination probability, (b) p_o, new call blocking probability, (c) p_{nc}, total call blocking probability.

variance has become finite and the tail has shrunk. Finally, for α equal to 100.0, there is almost no variance and the distribution has become, for the most part, deterministic about the mean. Another important aspect about the Pareto distribution is that it never decays to 0. However, as shown in Figure 3, both the exponential and uniform distributions decay to 0 and have short tails.

Figure 4 shows effect of load on PCS blocking statistics for each of the different mobility distributions. η is fixed at 0.25μ. It is observed that the Pareto distribution family consistently yields slightly higher p_f, p_o and p_{nc} statistics, than the uniform or exponential and the uniform distribution yields slightly higher blocking statistics than the exponential.

Figure 5 is the same as the previous figure, except that η is equal to 2.50μ. Here, we observe a completely different behavior than that in Figure 4. The Pareto distribution with α equal to 5.00 or 100.0 yields p_f statistics that are twice those produced by the exponential and p_o and p_{nc} statistics that are almost three times the exponential statistics. Also, it is observed that the uniform blocking statistics are significantly higher than the exponential, which was not observed in the previous figure.

We believe the higher blocking statistics produced by the Pareto and uniform distributions in the above figures is due to the distributions causing more K-ary hand-offs that not only result in more force terminations, but become correlated with the newly arriving calls, resulting in a significant increase the new call blocking probability, p_o. However, more investigation is needed to fully understand this phenomenon.

Figure 6 is the same as the previous two figures, except that η is equal to 10.00μ. Unlike the first two figures, we see that the blocking statistics do not vary with respect to the mobility distribution. Also, all blocking statistics have decreased by an order of magnitude over the previous two figures.

To shed light on how the mobility rate effects the PCS blocking statistics across different mobility distributions, the PCS blocking statistics for each mobility distribution is shown as a function of η for different call workloads in Figures 7-9. It is observed from these figures that initially as η increases, the blocking statistics for all the distributions increase (some more than others). However, when η is 2.00μ, the blocking statistics are starting to decrease. We also observe that the variance in blocking statistics for the different mobility distributions ceases for $\eta > 5.00\mu$. This trend appears across different call workloads (λ equal to 1.25, 3.75 and 6.25 Erlangs). We believe the initial increase in blocking statistics as η increases is due to (i) an increase in K-ary hand-offs and (ii) these hand-off being correlated with the arrival of new calls so that many of these new calls are blocked and the hand-offs are forced to terminate. However, as η continues in increase this correlation ceases and the high mobility rate actually helps to stabilize the system by allowing portables to move to less busy cells with the result being that new arriving calls are able to connect.

4 CONCLUSIONS

In this paper we present a model for simulating large-scale PCS networks that takes into account population dependent mobility patterns. We reduce the wall clock execution time of this computationally intensive simulation model without compromising the statistical integrity of the model by using a Time Warp distributed simulator. The distributed simulator has been shown to reduce the execution time of PCS simulations from 20 hours down to 3.5 hours. Using this distributed simulator, we study the effect of different population dependent mobility patterns and call workloads on PCS network performance. To generate different mobility patterns, the mobility distribution function was varied across 5 different probability distributions: (i) exponential, (ii) Pareto with α equal to 1.6, (iii) Pareto with α equal to 5.0, (iv) Pareto with α equal to 100.0 and (v) uniform. From this preliminary study, the primary results are:

- The Pareto distribution for medium mobility rates ($0.75\mu < \eta < 2.50\mu$) appears to yield significantly higher PCS blocking statistics than the exponential, especially when the Pareto distribution is deterministic ($\alpha = 100.0$).
- For high mobility rates ($\eta > 5.0\mu$), PCS blocking statistics appears to decrease. However, because only a few higher mobility rates are covered in this study ($\eta = 5.0\mu$, 10.0μ, and 15.0μ), it is possible that PCS blocking statistics cycle up and down as the mobility is increased. Further study is needed.

Finally, areas such as hand-off strategies, dynamic channel assignment, and wireless packet switching can be investigated by extending the simulation model presented here. These are topics of future work.

REFERENCES

Brown., R. 1988. Calendar Queues: A Fast O(1) Priority Queue Implementation for the Simulation

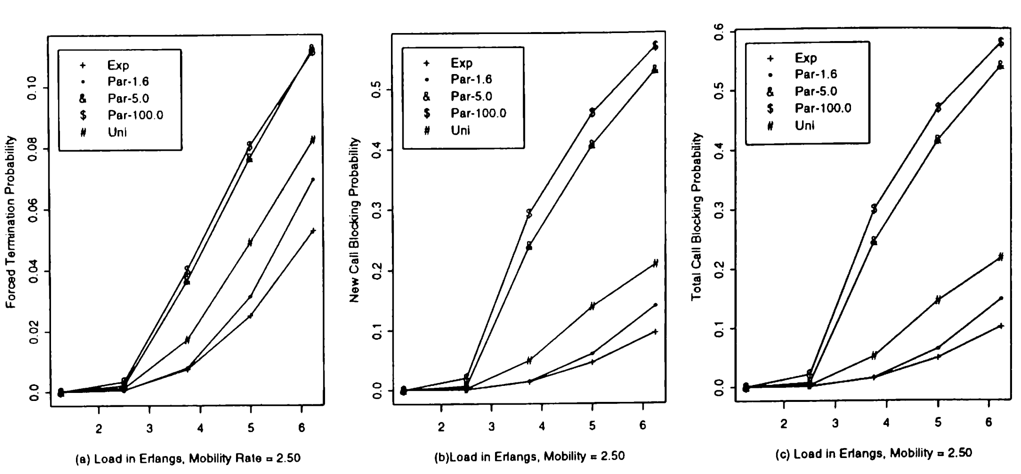

Figure 5: Effect of exponential, Pareto, and uniform mobility distributions on blocking probabilities as a function of λ in Erlangs. $\eta = 2.50\mu$. (a) p_f, forced termination probability, (b) p_o, new call blocking probability, (c) p_{nc}, total call blocking probability.

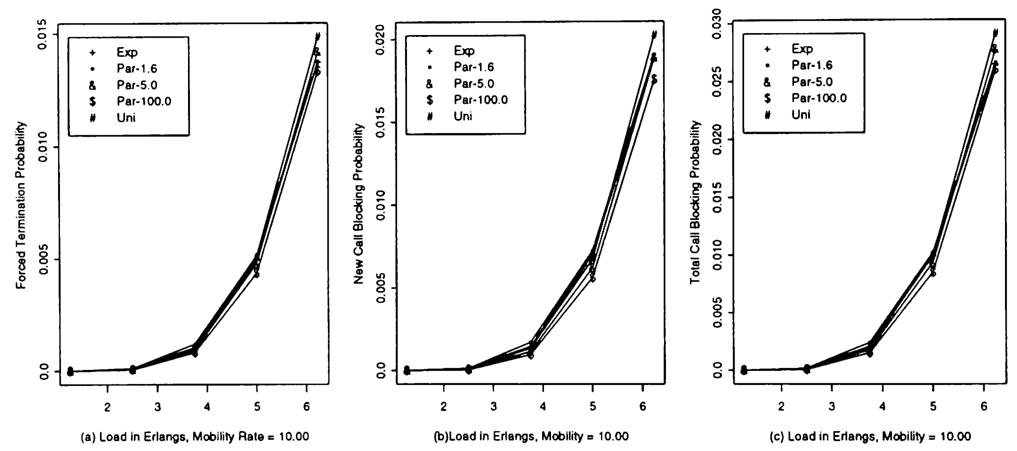

Figure 6: Effect of exponential, Pareto, and uniform mobility distributions on blocking probabilities as a function of λ in Erlangs. $\eta = 10.00\mu$. (a) p_f, forced termination probability, (b) p_o, new call blocking probability, (c) p_{nc}, total call blocking probability.

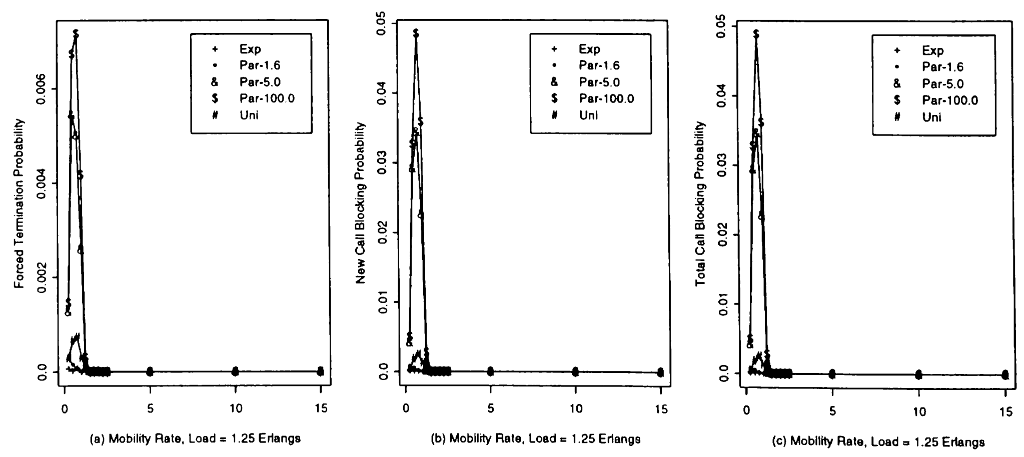

Figure 7: Effect of exponential, Pareto, and uniform mobility distributions on blocking probabilities as a function of η. $\lambda = 1.25$ Erlangs. (a) p_f, forced termination probability, (b) p_o, new call blocking probability, (c) p_{nc}, total call blocking probability.

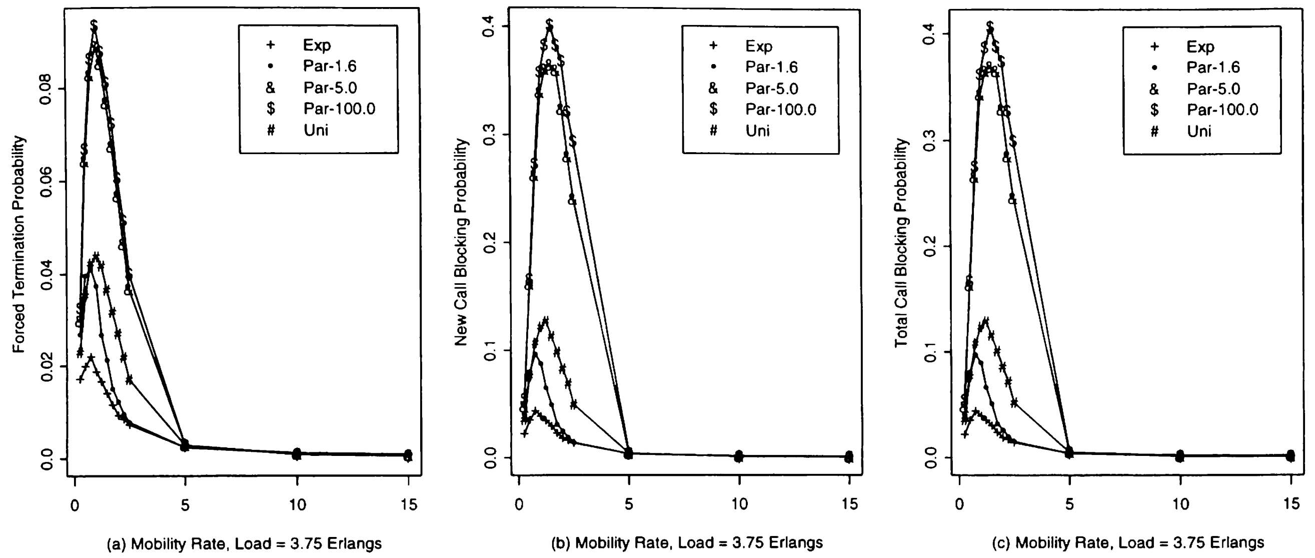

Figure 8: Effect of exponential, Pareto, and uniform mobility distributions on blocking probabilities as a function of η. $\lambda = 3.75$ Erlangs. (a) p_f, forced termination probability, (b) p_o, new call blocking probability, (c) p_{nc}, total call blocking probability.

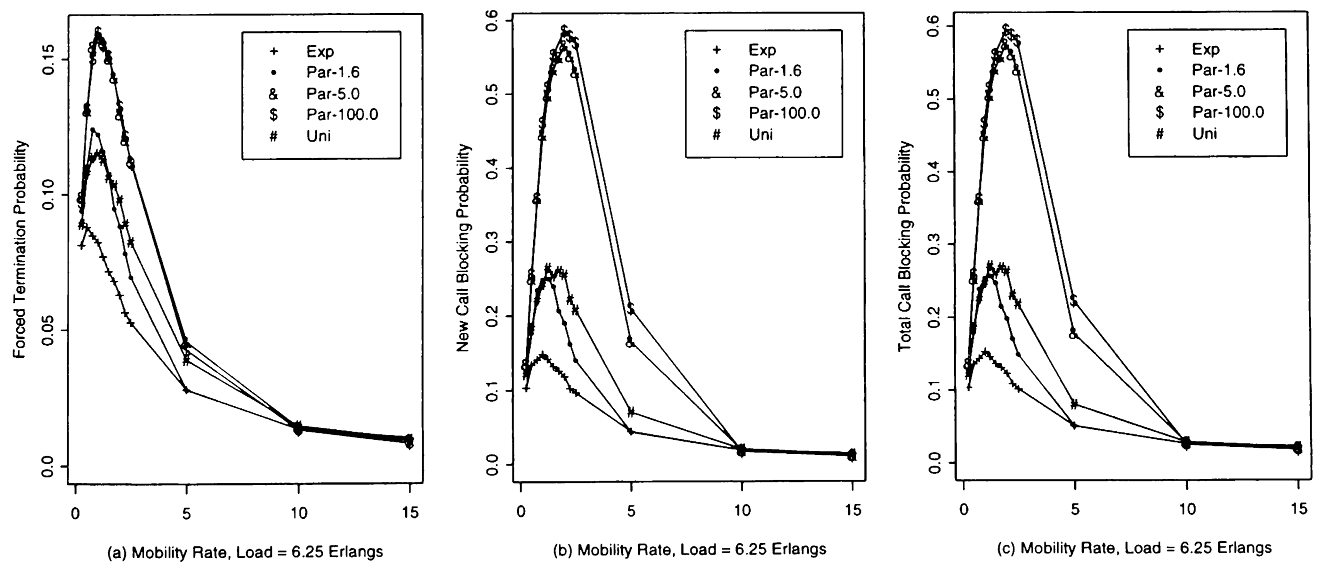

Figure 9: Effect of exponential, Pareto, and uniform mobility distributions on blocking probabilities as a function of η. $\lambda = 6.25$ Erlangs. (a) p_f, forced termination probability, (b) p_o, new call blocking probability, (c) p_{nc}, total call blocking probability.

Event Set Problem. *Communications of the ACM*, 31(10), pp. 1220–1227, October.

Carothers, C., Fujimoto, R.M., and Lin, Y.-B. 1995. A Case Study in Simulating PCS Networks Using Time Warp. In *Proceedings of the 9th Workshop on Parallel and Distributed Simulation*, pp. 87–94, June.

Carothers, C., Fujimoto, R.M., Lin, Y.-B., and England, P. 1994. Distributed Simulation of PCS Networks Using Time Warp. *Proceedings of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pp. 2–7, Feburary.

Chuang, J. C-I. 1993. Performance Issues and Algorithms for Dynamic Channel Assignment. *IEEE Journal on Selected Areas in Communications*, 11(6), pp. 955–963, August.

Cox, D. C. 1995. Wireless Personal Communications: What Is It? *IEEE Personal Communications*, pp. 20–35, April.

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

Fujimoto, R. M. and Nicol, D. M. 1992. State of the Art in Parallel Simulation. *Proceedings of the 1992 Winter Simulation Conference*, pp. 122–127, December.

Geist, A., et al. 1993. PVM 3 User's Guide and Reference Manual. Technical Report TM-12187, Oak Ridge National Laboratory, May.

Jefferson, D. R. 1985. Virtual Time. *ACM TOPLAS*, 7(3), pp. 404–425, July.

Lin, Y.-B., and Mak, V.K. 1993. On Simulating a Large-Scale Personal Communications Services Network. To appear in *ACM Transactions on Modeling and Computer Simulation.*

Misra, J. 1986. Distributed-Discrete Event Simulation. *ACM Computing Surveys*, 18(1), pp. 39–65, March.

Richter, R. and Walrand, J. C. 1989. Distributed Simulation of Discrete Event Systems. *Proceedings of the IEEE*, 77(1), pp. 99–113, January.

Tekinary, S., and Jabbari, B. 1992. A Measurement Based Prioritization Scheme for Handovers in Cellular and Microcellular Networks. *IEEE Journal on Selected Areas in Communications*, 10(1), pp. 1343–1350, October.

Wong, W. C. 1993. Packet Reservation Multiple Access in a Metropolitan Microcellular Radio Environment. *IEEE Journal on Selected Areas in Communications*, 11(6), pp. 918–925, August.

AUTHOR BIOGRAPHIES

CHRISTOPHER CAROTHERS is a Ph.D. student in the College of Computing at the Georgia Institute of Technology. He received a BS degree in Computer Science (1991) from the Georgia Institute of Technology in Atlanta, Georgia. His research interests include parallel and distributed simulation and the performance modeling of personal communications services networks.

RICHARD FUJIMOTO is a professor in the College of Computing at the Georgia Institute of Technology. He received BS degrees in Computer Science (1977) and Computer Engineering (1978) from the University of Illinois in Urbana-Champaign, and MS (1980) and Ph.D. (1983) degrees from the University of California in Berkeley. He has been actively engaged in research in parallel and distributed simulation since 1986. He is currently an area-editor for ACM TRANSACTIONS ON MODELING AND COMPUTER SIMULATION, and chairs the steering committee for the annual WORKSHOP ON PARALLEL AND DISTRIBUTED SIMULATION (PADS).

YI-BING LIN received his BSEE degree from National Cheng Kung University in 1983, and his Ph.D. degree in Computer Science from the University of Washington in 1990. Between 1990 and 1995, he was with the Applied Research Area at Bell Communications Research (Bellcore), Morristown, NJ. In 1995, he was appointed full professor of Department and Institute of Computer Science and Information Engineering, National Chiao Tung University. His current research interests include design and analysis of personal communications services networks, distributed simulation, and performance modeling. He is a subject area editor of the JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING, an associate editor of the INTERNATIONAL JOURNAL IN COMPUTER SIMULATION, an associate editor of SIMULATION magazine, a member of the editorial board of INTERNATIONAL JOURNAL OF COMMUNICATIONS, a member of the editorial board of COMPUTER SIMULATION MODELING AND ANALYSIS, Program Chair for the 8th WORKSHOP ON DISTRIBUTED AND PARALLEL SIMULATION, and General Chair for the 9th WORKSHOP ON DISTRIBUTED AND PARALLEL SIMULATION.