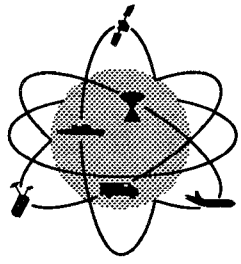
TECHNICAL RESEARCH REPORT

Efficient Simulation of DEDS by Means of Standard Clock Techniques: Queueing and Integrated Radio Network Examples

by J.E. Wieselthier, C.M. Barnhart, and A. Ephremides

CSHCN TR 93-11/ISR TR 93-99



CENTER FOR SATELLITE & HYBRID COMMUNICATION NETWORKS

A NASA CENTER FOR THE COMMERCIAL DEVELOPMENT OF SPACE

University of Maryland Institute for Systems Research

Naval Research Laboratory
Washington, DC 20375-5320

NRL/MR/5521--93-7392

Efficient Simulation of DEDS by Means of Standard Clock Techniques: Queueing and Integrated Radio Network Examples

JEFFREY E. WIESELTHIER
CRAIG M. BARNHART

Communication Systems Branch
Information Technology Division

ANTHONY EPHREMIDES

Locus, Inc.
Alexandria, Virginia
and
University of Maryland
College Park, Maryland

September 7, 1993

CONTENTS

EFFICIENT SIMULATION OF DEDS BY MEANS OF STANDARD CLOCK TECHNIQUES: Queueing and Integrated Radio Network Examples

1 INTRODUCTION

Simulation is an important tool in the study of communication networks, manufacturing systems, and other complicated man-made systems. Many systems of this type are examples of discrete-event dynamic systems (DEDS) for which analytical models have not yet been developed; thus simulation is the primary method for their performance evaluation and control. Although in many cases simulation can provide a good estimate of system performance, it can also be extremely time consuming, and therefore expensive. This is especially true when it is necessary to evaluate system performance for a large number of values of one or more parameters or for different control policies.

Recent research in the area of DEDS has developed a number of approaches that greatly improve the efficiency of the simulation process. For example, a number of perturbation analysis (PA) methods have been developed that can provide sensitivity information (and hence a characterization of system performance under a wide range of parameter values) from the observation of a single simulation run [1-5]. In contrast, the traditional "brute force" approach to simulation requires a separate simulation run for each parameter value of interest. The most basic PA method is Infinitesimal Perturbation Analysis (IPA), which is applicable only when the performance metric is continuous in the parameters of interest. However, other PA methods have been developed to accommodate a wide variety of problems in which discontinuities are present, e.g., finite perturbation analysis (FPA) [6, 7], extended perturbation analysis (EPA) [3, 8], and smoothed (conditional) perturbation analysis (SPA) [5, 9, 10].

One particular variation of the PA approach that is applicable to systems with discrete parameters is the *Standard Clock* (SC) technique developed by Vakili et al. [11, 12] and used by Cassandras et al. for networking examples [13]. The primary advantages of this method are that it is relatively easy to implement, and that it provides considerable improvement in simulation efficiency. Furthermore, it can also be used for on-line optimization and control. A disadvantage of this approach is that it is usable only when interevent times are exponential; however, many systems can be sufficiently well characterized by exponential processes, so that this is not necessarily a serious limitation. Moreover, it was recently shown by Ho et al [14] that certain deterministic events can be incorporated into the SC model, and we in fact have extended

Manuscript approved June 25, 1993.

the SC model to incorporate the modeling of fixed-length data packets in an integrated voice/data network.

In this report we describe the SC approach, and demonstrate its application to several examples of DEDS. First we consider the M/M/1/K queue, and demonstrate the improved simulation efficiency that can be achieved using this approach. In this case we are interested in system performance for many values of the discrete parameter K (the buffer size). The availability of an exact analytical solution for this queueing model permits the validation of our SC simulation model. In addition to the study of a single M/M/1/K queue, we study networks of interconnected queues of this type. Such queues are chosen because they are nontrivial, but relatively simple, objects on which the power of the SC technique can be tested.

Although the SC approach is ideally suited to parallel computing, performance improvement can also be achieved on a sequential computer, and our studies are based primarily on the use of sequential platforms. In our study of queueing examples, our focus is on the improvement in simulation times that can be achieved, rather than on the performance results of the system being simulated. We identify and track the time spent in the various aspects of the simulation, including random number generation and state updating, and compare predicted performance improvement with that which is actually measured. Although a number of papers have been written on the SC approach, they do not address in detail the amount of time spent in each of the components of the simulation. The explicit use of the simulation time breakdown permits a judicious assessment of the expected improvement in a given application. The results of our study, including detailed simulation time breakdowns, have also appeared in [15, 16].

We then extend the use of SC simulation to examples that involve circuit-switched voice networks. The applicability of an exact "product-form" analytical solution for the state distribution again permits the validation of our simulation model. Here the discrete parameter set is the admission-control policy that determines which voice calls are to be accepted and which are to be blocked. Simulation timing results for this network indicate that the SC approach should scale well when applied to complex problems. However, our primary focus shifts from such efficiency results to actual measures of the network performance and performance optimization. Finally, we "combine" the network-of-queues model and the circuit-switched voice network model to yield a SC simulation of an integrated voice and data network.

Performance results are presented to address many aspects of system operation. The accuracy of our voice-traffic simulation model is demonstrated by the fact that the difference between simulated and exact results for our sample network is less than 0.2% for each of 120 control policies that have been studied. Furthermore, we demonstrate that the SC approach is useful for "ordinal optimization," i.e., the determination of some of the best (although not

necessarily the optimal) policies on the basis of relatively short simulation runs. This approach is effective for both voice and data performance metrics.

A crucial observation from our ordinal optimization studies, and one that may have interesting and possibly far-reaching implications in the study of optimization methods, is that crude models are often adequate to predict the relative performance of different control policies. For example, we demonstrate that exceptionally accurate policy rankings can be obtained from simulation models that incorporate key aspects of the systems being modeled, although they do not incorporate all aspects of system operation and do not predict performance (e.g., delay) accurately. Furthermore, we have also obtained highly accurate policy rankings using a crude analytical model for delay performance. The accuracy of the rankings obtained in this manner suggests that simple analytical models can be used to reduce the search space significantly to just a few policies (perhaps to just a single one) whose performance can then be evaluated accurately via simulation, thus decreasing computation time dramatically.

1.1 Outline of the Report

In Section 2 we discuss the M/M/1/K queue, and in Section 3 we use it as an example to explain the SC simulation methodology. In Section 4 we present our simulation results for the SC simulation of M/M/1/K queues. We first demonstrate the validity of our simulation model by comparing simulated results with the well-known analytical model. We then proceed to demonstrate the improvement in efficiency that is possible through the use of the SC approach. In addition to quantifying the overall improvement, we identify and track the time spent in the various aspects of the simulation, and demonstrate that there is a theoretical upper bound on the speedup attainable by the use of SC simulation on a sequential machine. In Section 5 we extend our model to networks of queues, and we demonstrate that the efficiency of the SC method scales well with problem complexity.

In Section 6 we develop an SC simulation model for circuit-switched voice radio networks. The network model is based on the one we studied earlier in an analytical and computational study of admission-control policies in networks of this type [17]. SC simulation timing results again show that the SC approach scales well with problem size. In Section 7 we extend our SC simulation model of the previous section to the case of integrated networks that incorporate data traffic (with fixed-length single-packet messages) as well as voice. In Section 8 we present extensive performance results from SC simulations, from which we make a number of interesting and significant observations. First, we demonstrate the validity of the simulation of the voice-traffic process by showing excellent agreement with the theoretical product-form model. In addition, we discuss the principles of ordinal optimization and demonstrate the use of

such techniques to find good solutions from relatively short simulation runs. Also, we apply the concepts of ordinal optimization to the data-traffic process as well, and demonstrate that simple analytical models (although poor at predicting actual delay values) are useful for finding good control policies.

Finally, in Section 9 we present our conclusions from this research.

2 THE M/M/1/K QUEUE

The first system we simulated using the SC approach was the M/M/1/K queue, a well-understood system for which the equilibrium solution is available in a simple closed form. It is precisely because we know the analytical solution for the M/M/1/K queue that it is a good, simple problem for the development and illustration of simulation models. Cassandras has, in fact, used it for that purpose in [5]. In this section we describe the M/M/1/K queueing model and present its analytical solution.

An M/M/1/K queue, such as the one shown in Fig. 1, is a single-server queue with finite buffer capacity K (including the packet in service), Poisson arrival process at rate λ, and exponentially distributed service of expected duration $1/\mu$. We use the following notation to describe the state of the queueing system:

x = the number of packets in the system (including the packet currently being served, if any),

$\pi(x)$ = the steady-state probability that there are x packets in the system.

As a result of the finite buffer capacity, arrivals are blocked and lost from the system when the buffer is full, i.e., when $x = K$. Thus, the fraction of arrivals that are blocked is $\pi(K)$, and the fraction that is accepted is $1-\pi(K)$, as indicated in the figure.

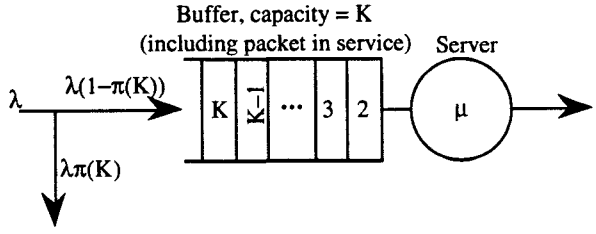


Fig. 1 — An M/M/1/K queue

The steady-state queue occupancy distribution is [18]

$$\pi(x) = \begin{cases} \dfrac{(1-\rho)\rho^x}{1-\rho^{K+1}} & 0 \le x \le K \\ 0 & \text{otherwise} \end{cases},$$

where $\rho = \lambda/\mu$. Note that $\rho \geq 1$ does not result in instability in this type of queueing system because excess traffic is blocked from entering the system. The finite buffer capacity effectively "thins" the traffic seen by the server so that as ρ approaches infinity, the actual server utilization rate approaches 1. The availability of an analytical model for the M/M/1/K queue has permitted us to verify the accuracy of our simulations. In particular, we have simulated the case of $\rho = 1$, for which it is easy to show (by application of L'Hopital's rule) that the queue length distribution is uniform, i.e., $\pi(x) = 1/(1+K)$, $x = 0, 1, \ldots, K$.

3 THE STANDARD CLOCK APPROACH TO SIMULATION

The principles of SC simulation are best described by means of an example, and in this section we use the M/M/1/K queue to explain the SC methodology. In later sections we apply SC techniques to other examples as well.

We have noted that the availability of an analytical solution for the M/M/1/K queue makes it a good, simple problem for the development and illustration of simulation models. The presence of the discrete parameter K, the buffer capacity, raises the issue of the generation of simulation results for a large number of values of K. Under "traditional" or "brute force" methods, it would be necessary to perform K simulations, i.e., one for each value of this parameter. A number of perturbation analysis (PA) methods have recently been developed that can describe system performance under a wide range of parameter values from the observation of a single simulation run. However, most perturbation analysis techniques that are effective in estimating gradients for continuous variables generally do not work with discrete variables. One exception to this limitation is the SC simulation technique, which works equally well for continuous or discrete variables. Hence, the discrete nature of the parameter K makes the M/M/1/K queue a particularly good candidate for SC simulation.

We should mention that in certain cases other PA methods (e.g., IPA) have been modified to work for a particular problem by using a form of "smoothed perturbation analysis" (SPA) [5, 9, 10], which takes the conditional expectation over a family of discontinuous sample paths, resulting in a continuous function that is amenable to analysis. One particular application of SPA that has been applied to networking problems is Cassandras et al.'s use of "marking" and "phantomizing" techniques for the generation of link-activation schedules [19, 20].

In SC simulations of an M/M/1/K queue, rather than generating separate timing sequences for each event type (i.e., arrivals and departures) as is typically done in traditional

simulations, a single sequence of random numbers is generated from an exponential distribution[1] with parameter $\Lambda = \lambda + \mu$. Each of the variates in this sequence represents a generic interevent time. The type of event (i.e., arrival or departure) associated with each interevent time is determined by drawing an independent random number U from a uniform distribution on [0, 1], and declaring

$$\text{event} = \begin{cases} \text{arrival} & U \le \lambda/\Lambda \\ \text{departure} & U > \lambda/\Lambda \end{cases}.$$

Thus two random numbers must be generated to produce each event, one to specify the timing of the next event and the other to specify its type. The event is an arrival with probability λ/Λ or a departure with probability μ/Λ. It is possible that an event determined in this manner turns out to be infeasible (e.g., a departure from an empty system). The interevent time of such a "fictitious" event is used to update the system time as if the event were "real" and did in fact occur, but no state change occurs (the fictitious event is discarded).

The efficiency of the SC method is obtained by using the resulting sequence of (interevent time, event) pairs, which is known as the *clock sequence*, to simultaneously generate sample paths for a number of structurally similar, but parametrically different, systems. In particular, a single clock sequence can be used to generate N sample paths in parallel for N M/M/1/K queues, i.e., K ranges from 1 to N. Arrival events are always feasible for all values of K; however, an arrival when $x = K$ does not result in a change of state. Similarly, a fictitious event, e.g., a departure when $x = 0$ in this system, leaves the state unchanged. Thus a different trajectory may be produced in each of these experiments, even though the clock sequence is the same for all. Since each element of the clock sequence is used N times, the number of events that must be generated is reduced by a factor of N. Improved simulation efficiency results from the fact that the generation of events is considerably more time consuming than the consequent updating of system state, as our simulation timing results show (see Section 4). Use of the SC approach permits the event sequence generated for a single experiment to be used for a large number of parallel experiments, thereby eliminating the need to redo the most time-consuming aspect of simulation for each experiment.

A disadvantage of the SC approach is that infeasible events can be generated, thus resulting in wasted computational effort. The fraction of events that are infeasible depends strongly on the system that is being simulated, and in our studies represented a small fraction of the events that were generated.

---

[1] The SC approach is applicable only when the interevent times are exponential, although we demonstrate in Section 7.2 how certain deterministic events (which in our example are fixed-length data packets) can be incorporated into the simulation.

Another limitation of this approach is that it is restricted to systems with exponential interarrival times, although it has recently been shown that some deterministic events can also be incorporated into the model [14], and we, in fact, do so in our development of the integrated network model in Section 7. Actually, many queueing systems (and presumably many other DEDS as well) are not very sensitive to the requirement of exponential service times, although the requirement for exponential interarrival times (i.e., a Poisson arrival stream) is more critical. For example, the equilibrium state occupancy distribution for the M/M/c/c queue[2] (which is characterized by the well-known Erlang loss formula [21]) is also applicable to the M/G/c/c queue, i.e., the distributions are identical provided that the service time distributions have the same mean. Also, it has been shown that performance of circuit-switched voice networks with Poisson arrival statistics is relatively insensitive to the service time distribution [22]. Thus, in examples like these, it is possible to evaluate performance based on the assumption of exponential service times, and then apply these results to systems with more-general characteristics.

The applicability of the SC approach is not limited to simulation examples. It can also be used in conjunction with *on-line system observation* to predict the sample paths that would be generated under many different sets of parameters, based on the observation of a single sample path under the *nominal* set of system parameters in an actual system [13]. Under this approach, the event sequence and state sequence of the nominal system is observed. Based on knowledge of system parameters in the nominal system and in each of many perturbed systems, some of the events are transformed into events of different types (e.g., an arrival into a departure). The perturbed event sequence results in a perturbed trajectory, corresponding to each new set of system parameters. The event-transformation process is complicated by the fact that some events may not be observable in the nominal system. For example, when the nominal system is empty, no departure events will be generated, although in a simulation of the system the SC approach would generate them (and ignore them as fictitious events). It is necessary to generate the events that would be ignored as fictitious in the nominal system because they may be needed by some of the perturbed trajectories. The SC approach provides a method to generate such unobservable events with appropriate probabilities [13]. On-line observation techniques such as this one can be used to determine optimal control policies. Thus the SC method is considerably more powerful than conventional simulation techniques, which are limited to the evaluation of system performance, rather than the actual control of on-line systems. We have not yet exploited the

---

[2] The notation M/M/c/c refers to a standard queueing system with exponential (Markovian) interarrival and service times, c servers, and a maximum of c customers permitted in the system; thus there is no queueing and the customers that cannot be accommodated at the time of their arrival are blocked and lost from the system. The M/G/c/c system is identical, except that the service time has a general distribution.

capability of the SC approach for such on-line system control, but we expect to do so in the near future.

Figure 2 shows flow charts of two programs we have used to simulate M/M/1/K queues. The *Brute-Force* (BF) simulation program uses the conventional approach of sequentially simulating each of the N queues for the entire length of their sample paths. It serves as our benchmark for measuring the efficiency of the SC simulation method. The Standard Clock simulation program simulates all N queues in parallel; each generated event is passed to all N queues so that the number of events generated is reduced by a factor of N.

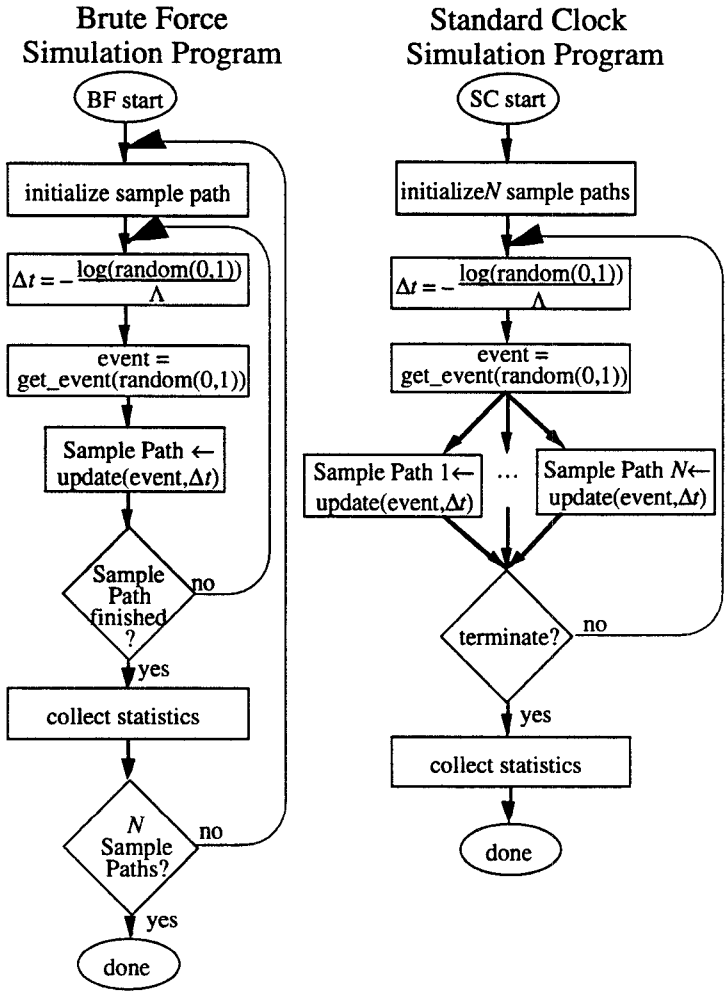


Fig. 2 — Flow charts for the BF and SC simulations

A comment is necessary on the approach we used to generate the BF simulation results. Instead of performing a true BF simulation, in which separate event streams would be generated for each event type, we actually performed a separate SC simulation for each value of K. Thus our BF simulation consisted of N SC simulations (each for a single value of K) performed sequentially. Therefore, in our simulations of M/M/1/K queues, both the SC and the BF runs

must send approximately the same number of events to each sample path. Our use of the SC approach to model BF simulations results in somewhat reduced efficiency, because true BF simulations do not generate fictitious events, whereas SC simulations do. However, the SC approach eliminates the need to maintain and check a state-dependent feasible event set that would normally be needed in BF simulations. Furthermore, since the number of fictitious events in our simulations (for $\rho = 1$) is quite small (and decreases as K increases), their generation does not significantly penalize our timing measurements for the BF approach. The timing error introduced by our use of this approach in BF simulation is, in fact, insignificant in comparison with the degree of improvement that is achieved by using the SC simulation method.

The four primary functions, namely `random`, `log` (i.e., the natural log), `get_event`, and `update`, are exactly the same in both the BF and the SC programs. The function `random` is a non-linear additive feedback random number generator that uses a default table to return integer random numbers from a uniform distribution on $[0, 2^{31} - 1]$. Its period is approximately $16 \times (2^{31} - 1)$ [23].

The `get_event` function uses its argument U, a real-valued random number drawn from a uniform distribution on [0, 1] (i.e., U = random(0,1) = `random`/MAX_INT, where MAX_INT = $2^{31} - 1$), to navigate through a pair of tables and determine the event type. The construction of these tables, which is performed during the initialization phase of the simulation, and their use to efficiently determine the mapping of the random number U to a specific event, is collectively known as the *alias method* [24].

For problems as simple as the M/M/1/K queue, which only has two event types (arrivals and departures), the use of the alias method is not really necessary since we only need to flip a weighted coin to determine the event type. However, the use of the alias method in more-complex problems that have numerous event types can result in significant simulation time savings by expediting event-type determination. Since our intent is to use SC simulation to analyze such problems, we have used the alias method in this simple example to provide a baseline for comparison with more-complex problems that can truly benefit from its use.

In more-complex simulations where the number of events and the frequency of their occurrence varies with the sample path parameters, the use of custom alias tables for each sample path (see Section 6.2) may increase the efficiency of event usage (i.e., reduce the number of fictitious events) sufficiently to compensate for the added overhead of their creation. However, because each of the two types of events occurs with fixed probability, regardless of the sample path and K value, creating new alias tables for each sample path does not increase the efficiency when simulating M/M/1/K queues. Therefore, we have made the BF program as efficient as

possible by using a single pair of alias tables (one-time setup cost) for all sample paths, rather than setting up new ones for each sample path.

The function `update` uses its two arguments, Δt and event, to move the simulation clock forward by Δt time units and update the system state to reflect the effect of the event. The state update function for the $(n+1)^{st}$ event is

$$x(n+1)=\begin{cases} x(n)+1, & \text{event = arrival, } x(n)<K \\ x(n), & \begin{cases}\text{event = arrival, } x(n)=K \\ \text{event = departure, } x(n)=0\end{cases} \\ x(n)-1, & \text{event = departure, } x(n)>0 \end{cases}$$

4 SC SIMULATION RESULTS FOR M/M/1/K QUEUES

The SC and the BF simulation models for M/M/1/K queues were programmed in C++ and run on a Sun-4/330 workstation. Each run of one of these programs simulates N different M/M/1/K queues with different buffer capacities, i.e., $K = 1, \ldots, N$. Thus, in the SC simulations N different sample paths are generated in parallel (on a sequential machine).

In Section 4.1 we demonstrate the accuracy of the SC simulation approach by comparing the statistics of the simulated processes with their theoretical values. However, in Section 4.2 we shift our attention to the speedup achievable by using the SC approach as compared to BF simulation. Finally, in Section 4.3 we briefly address the issues associated with SC simulation on parallel machines.

4.1 SC Queueing Statistics and Convergence Properties

Figure 3 shows how the buffer occupancy distribution from one realization of an M/M/1/K queue with $K = 10$ and $\rho = 1$ converges towards the theoretical value of $1/(K+1) = 0.909$. (Recall that when $\rho=1$, $\pi(x) = 1/(K+1)$ for $x = 0, 1, \ldots, K$.) In this figure, each curve represents one of the values of x; the curves are not labeled because all we are interested in here is how close the curves are to the theoretical value, which is the same for all of them.

To determine the distribution $\pi(x)$ we have taken advantage of the fact that *Poisson arrivals see time averages* (the so-called PASTA property). As a result of this property, it is sufficient to monitor the queue size at arrival instants to determine the state distribution over all time; it is unnecessary to examine the system state over the continuum of time, e.g., by monitoring the time spent in each state. Also, note that the distributions shown in the curves in Fig. 3 represent the average state occupancies over the entire simulation up to that point, rather than a moving or weighted average over the most recent portion of the simulation.

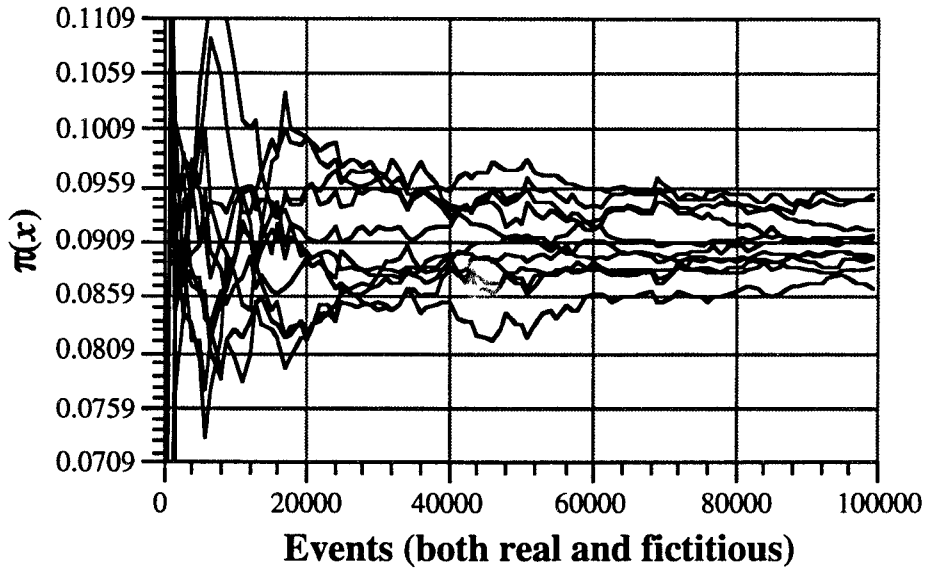


Fig. 3 — Sample path $\pi(x)$ vs. number of events; $K = 10$, $\rho = 1$

These results were obtained by using the Sun random number generator "random( )," and are typical of those obtained using any of a number of good random number generators, several of which were examined in our studies. Figure 4 shows the average of ten such runs, each of length corresponding to 100,000 events. Typically, the standard deviation of the value of $\pi(x)$ averaged over 10 different seeds (compared to the theoretical value) appears to converge toward a value of 0.001. Little improvement in the standard deviation is typically obtained by increasing the length of the simulations past about 30,000 events.

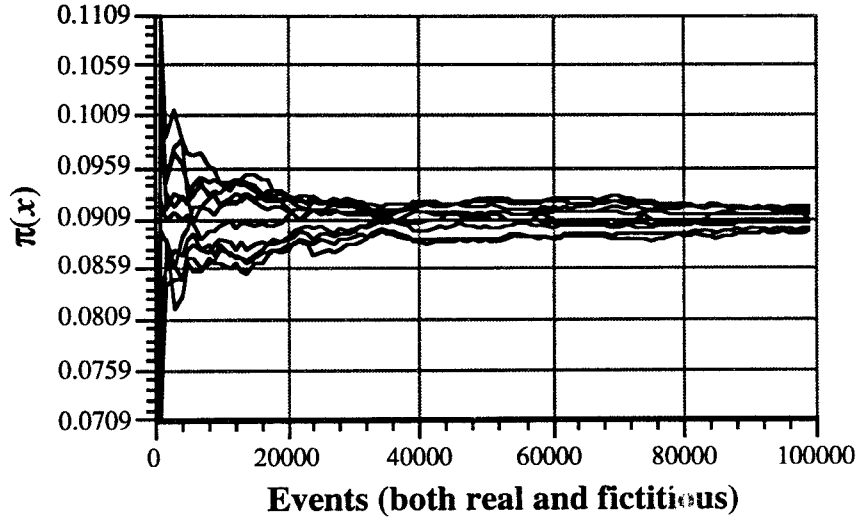


Fig. 4 —Sample path $\pi(x)$ vs. number of events; $K = 10$, $\rho = 1$, averaged over 10 seeds

4.2 SC Timing Results for M/M/1/K Queues

In Section 4.1 the accuracy of our SC simulation model was demonstrated; next we address the question of the improvement in simulation efficiency that is possible through the use of the SC approach.

Figure 5 compares the time required to simulate N sample paths using BF and SC methods as N varies from 1 to 1000. Cassandras presented similar timing results for SC and BF simulations of M/M/1/K queues for values of K from 3 to 10 in [5]. Our results differ from his in that we have studied much larger values of K, and in that we have also identified and tracked the time spent in various aspects of the simulation, as will be discussed shortly. Our detailed examination of simulation timing results has permitted us to develop an estimate of the improvement in simulation time that can be achieved using the SC approach on a sequential machine. In contrast, Cassandras studied only the total time spent in the simulations.

The timing results shown in Fig. 5 are based on averages taken over 10 runs using different random number generator seeds. The SC simulations were terminated when the queue with $K = 2$ had received 100,000 real events. It turns out that an average of 119,967.1 events (both real and fictitious) were required for these simulations. The BF simulations of each sample path were terminated after 119,967 events (real or fictitious).[3]

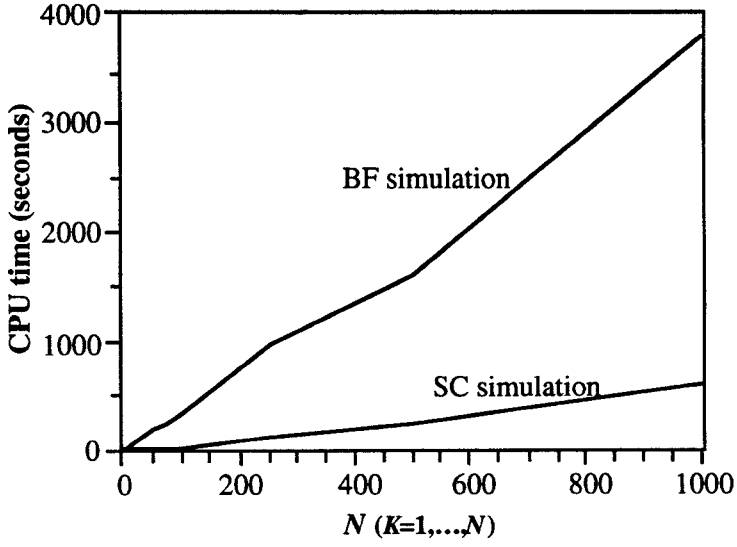


Fig. 5 — The time required to perform SC simulations and BF simulations

[3] Since the N sample paths are not generated in parallel in the BF simulations, and because the sequence of random variates is different for each sample path, we do not know how many events are required for a queue with $K = 2$ to see 100,000 real events. Therefore we use the average number of events (determined from 10 different SC simulations with different random number generator seeds) required for termination, namely 119,967.

We see in Fig. 5 that a significant time savings is achieved by using SC simulation instead of BF simulation, and that in both cases the total simulation time is roughly linear in N. The SC simulation of 1000 sample paths finished in about 10 minutes, nearly 54 minutes ahead of our BF simulation (a speedup by a factor of 6.62). The primary source of this time savings is the efficient use that the SC simulation makes of each event that is generated. By generating the sample paths in parallel, two calls to the random number generator (one for the event time and one to determine the event type) yields an event for each of the N sample paths.

By using the Sun performance analysis tool "gprof" [25], we were able to break down the simulation time spent in each of the major functions in the simulations. The results are shown in Table 1. Both runs were simulations of 100 M/M/1/K queues (i.e., K = 1, 2, ..., 100) with ρ = 1.0. The BF results are from a single simulation of the queues; the SC results are the average taken over 10 simulations (with different random number generator seeds) of the queues. The table shows the number of calls to each function, the total time in seconds spent satisfying these calls, the percentage of the total simulation time that was spent in each function, and the average time per function call.[4]

Table 1 — Timing breakdown for BF and SC simulations of 100 M/M/1/K queues

	BF simulation, N = 100, ρ=1				SC simulations, N = 100, ρ=1			
function	calls	time (s)	%time	time/call	calls	time (s)	%time	time/call
`update`	11,996,700	52.90	21.0	4.410μs	11,996,710	54.241	96.6	4.521μs
`get_event`	11,996,700	60.97	24.2	5.082μs	119,967.1	0.573	1.1	4.776μs
`random`	23,993,710	83.24	33.1	3.469μs	240,244.2	0.816	1.4	3.397μs
`log`	11,996,700	54.85	21.7	4.572μs	119,967.1	0.544	0.9	4.535μs
total event generation	11,996,700	199.06	79.0	16.592μs	119,967.1	1.933	3.4	16.105μs

The table is divided into two sections. The upper section, which consists solely of the `update` function, represents the time expense involved in actually manipulating a sample path, i.e., state updating and statistics collection. The lower section, which consists of the functions `get_event`, `random`, and `log`, represents the event generation process. The bottom line of the table shows the total event generation cost. The time per call in this line includes two calls to `random`.

[4] Although the times measured per function call are not identical, they are sufficiently close to verify the accuracy of our timing estimates.

Table 1 shows that the primary difference in computational requirements between BF and SC simulations is the number of events that must be generated. A BF simulation of 100 sample paths must generate 100 times as many events as a SC simulation of the same sample paths. Thus, the BF simulation spends more than three fourths of the total simulation time generating events, and the 52.9 seconds spent actually updating the sample paths and collecting statistics represents only 21% of the total simulation time. The 54.24 seconds used by the SC simulation for the update function is approximately the same as in the BF simulation, but here it is 96.6% of the total SC simulation time. Only 3.4% of the SC simulation is invested in event generation.

We define the simulation speedup to be

$$\text{speedup} = \frac{\text{BF simulation time}}{\text{SC simulation time}}.$$

The simulation speedup in this example is ((52.9+199.06)/(54.241+1.933)=) 4.485.

Focusing specifically on the time spent generating events, the speedup is (199.06/1.933=) 102.98, which is slightly better than the value of 100 that we expect. It is noteworthy, but perhaps not surprising, that this speedup is directly proportional to the number of sample paths simulated.

For the general case of simulating N sample paths, we estimate the *per-event* simulation speedup, denoted by $S_e(N)$, as

$$S_e(N) = \frac{\text{BF simulation time/event}}{\text{SC simulation time/event}} = \frac{N\left(t_{\texttt{update}} + t_{\texttt{get\_event}} + 2t_{\texttt{random}} + t_{\texttt{log}}\right)}{N t_{\texttt{update}} + t_{\texttt{get\_event}} + 2t_{\texttt{random}} + t_{\texttt{log}}}, \quad (1)$$

where $t_{\texttt{function}}$ is the time per function call. As noted earlier, our BF simulation is simply a SC simulation in which the individual sample paths are generated in sequence rather than in parallel. Therefore, in our simulations of M/M/1/K queues, both the SC and the BF runs must send approximately the same number of events to each sample path. In general however, because fictitious events need not be generated in the BF approach, a BF simulation can generate a sample path of given length with fewer events than a SC simulation of the same model. Thus, the per-event simulation speedup estimate of Eq. (1) is needed to accommodate differences in the number of events required in BF and SC simulations.

Now it is easy to see that as N increases, the per-event speedup achievable by using SC simulation on a sequential machine approaches the following limit.

$$\lim_{N\to\infty} S_e(N) = \frac{t_{\texttt{update}} + t_{\texttt{get\_event}} + 2t_{\texttt{random}} + t_{\texttt{log}}}{t_{\texttt{update}}}. \quad (2)$$

Since each of the functions listed in Table 1, as well as the "total event generation" category, are exactly the same in both the BF and the SC simulation programs, the time per function call should be exactly the same in both programs. Therefore we have taken the weighted averages of the time/call values shown in the table, expressed them in terms of a common denominator, namely the time per `log`, and substituted them into Eqs. (1) and (2) to obtain the predicted per-event simulation speedup,

$$S_e(N) = \frac{4.6095 N^{1/\texttt{log}}}{(3.6221 + 0.9874N)^{1/\texttt{log}}}, \tag{3}$$

and a theoretical estimate of the speedup attainable by using SC simulation of M/M/1/K queues with large N values on a sequential machine, i.e.,

$$\lim_{N\to\infty} S_e(N) = \frac{4.6095}{0.9874} = 4.6683.$$

Figure 6 compares the speedup found in simulations to that predicted by Eq. (3). It shows that the actual speedup obtained in simulations is much better than our estimate. Although this comparison of the predicted speedup with that found by simulation does not show particularly good quantitative agreement, it does show qualitatively that there is an upper bound on the speedup that is achievable when using SC simulation on a sequential machine, i.e., that the speedup does not continue to increase significantly once N has reached a sufficiently high value.

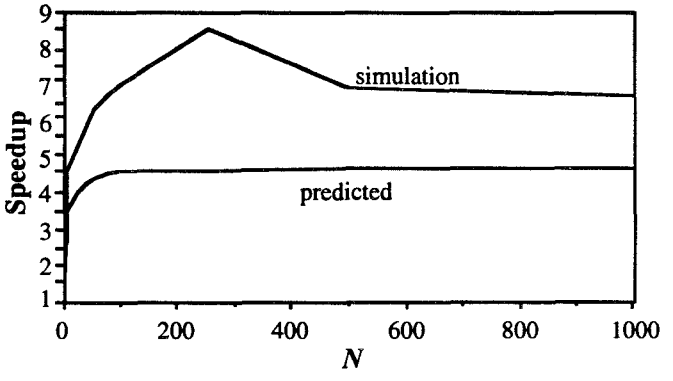


Fig. 6 — A comparison of the actual speedup vs. the predicted speedup

It is difficult to quantify precisely the timing in simulation runs, in part because the measurement tools disturb the quantities that they are measuring, and because of other processes concurrently active on the computer. Additionally, Eq. (3) accounts only for the four major functions; it neglects time spent in other portions of the simulation such as initialization and the "main" driver portion of the program. The simulation timing results include all aspects of the simulation, and are based on runs in which the measurement tools are not used. Thus they provide a more accurate measure of the speedup provided by the SC method.

4.3 SC Simulation on a Parallel Machine

Since the SC simulation method is based on the use of the same random number sequence to drive a number of experiments in parallel, it should be ideally suited for use on parallel machines. Figure 7 shows a conceptual data-flow model for simulation on such platforms.

We have developed, programmed and simulated this model in Sim++. The machines available for this work have been a network of four Sun workstations and an eight-processor Meiko Transputer. The results, in terms of simulation efficiency, have thus far been mixed. We have seen evidence that the model does exploit the inherent parallelism in the SC approach, but a number of different problems, primarily associated with memory usage and allocation, have thus far stymied efforts to scale up the problem to a size that would allow the benefits of the use of parallel simulation to be clearly demonstrated. In view of the availability of more-powerful sequential machines, we have not put a great deal of effort into the study of parallel machines. However, with the anticipated availability of a network of Sun Sparc-10 Workstations we expect to reexamine the use of parallel machines for SC simulation in the near future.

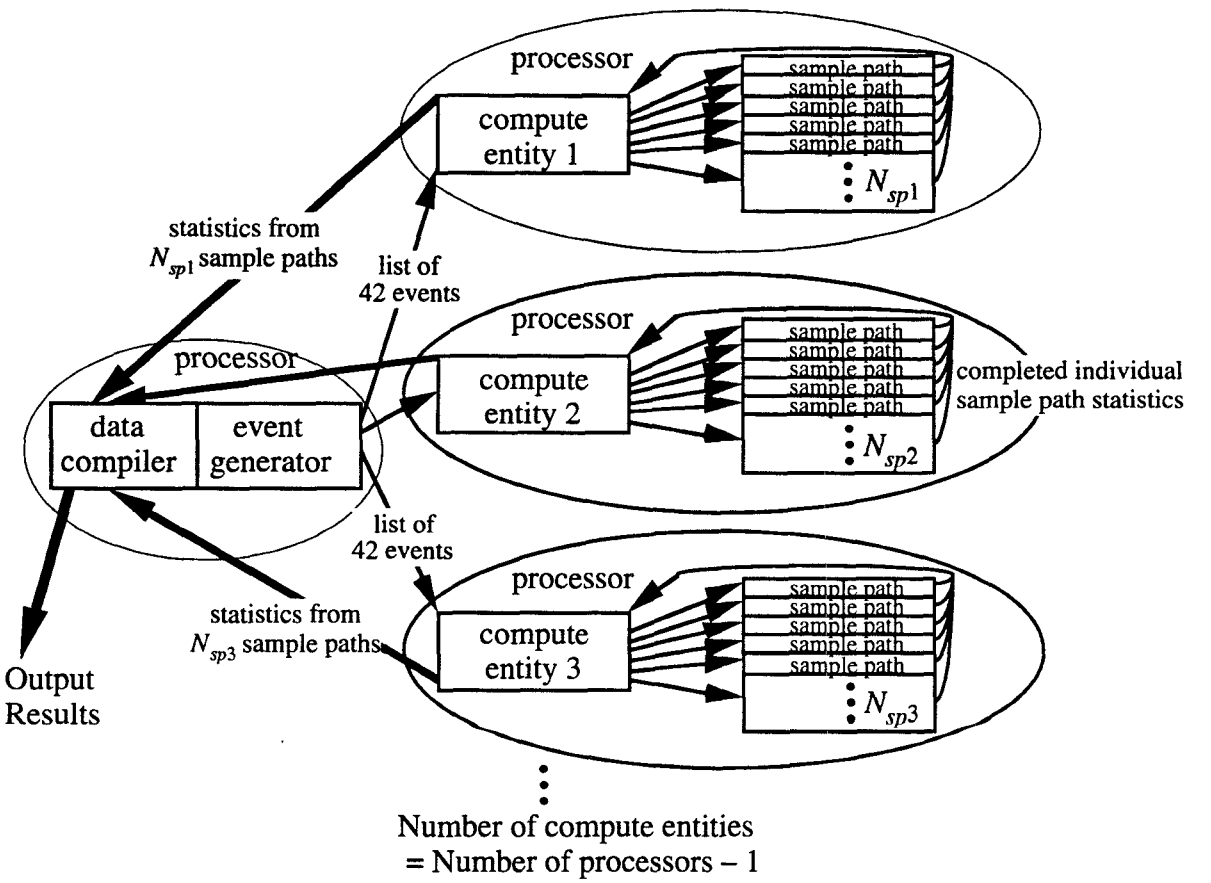


Fig. 7 — Data-flow model for parallel SC simulation

5 SC SIMULATION APPLIED TO NETWORKS OF M/M/1/K QUEUES

In this section, we extend the use of SC simulation to the examination of networks of M/M/1/K queues. Of course, once the M/M/1/K queues are interconnected, the arrival process at each of them is no longer Poisson (since the outputs of the queues are no longer independent). Therefore, we are actually studying networks of exponential servers with finite buffers. However, for conceptual purposes, and for notational convenience, we will refer to the system as a network of M/M/1/K queues. As we describe our network model, the reader will recognize that the model is a Jackson-like network with the exception that the buffers have finite capacity. Therefore, we expect that the independence assumptions used to make the Jackson network analytically tractable will become increasingly legitimate as the buffer capacities increase (i.e., as $K \rightarrow \infty$), and/or as traffic levels decrease (i.e., as $\rho \rightarrow 0$). As a result, the network statistics of a high capacity network with low traffic levels should also approach those of a true Jackson network. For a small network of J queues with low buffer capacities K, exact results can be obtained by solving the balance equations for the J-dimensional Markov chain with length $K+1$ in each dimension. However, this approach rapidly becomes impractical as either J or K increases.

5.1 The Network Model

Consider the network of queues shown in Fig. 8, which consists of four M/M/1/K queues that are fully interconnected. Queue i receives exogenous Poisson arrivals at rate r_i, has buffer capacity K_i, and it routes a departing packet to queue j with probability P_{ij}, or the packet exits the network with probability P_{ie}. Thus, along with its exogenous input, each queue receives a fraction, depending on the P_{ij}, of the (non-Poisson) output from the other queues (three, in this example); the total input rate to queue i is

$$\lambda_i = r_i + \sum_{j=1}^{J} \lambda_j \left(1 - \pi\left(K_j\right)\right) P_{ji}, \qquad 1 \le i \le J, \tag{4}$$

where the factor $(1-\pi(K_j))$ accounts for the traffic thinning that occurs at each queue as a result of the finite buffer capacity.

A standard Jackson network of J distinct exponential servers (with infinite buffers) is straightforward to analyze because the probability mass function of the system state has the product-form (see e.g., [18]). In this case the factor $(1-\pi(K_j))$ in Eq. (4) is not present, and the J equations are readily solved to yield the total arrival rate for each queue. However, for the case of finite-buffer queues, the presence of the factor $(1-\pi(K_j))$ greatly complicates the analysis; simulation appears to be the best, if not the only, approach for evaluating the network

performance. As mentioned previously, there are special cases (e.g., low traffic rates and/or high buffer capacities) where analytical approximations are possible.

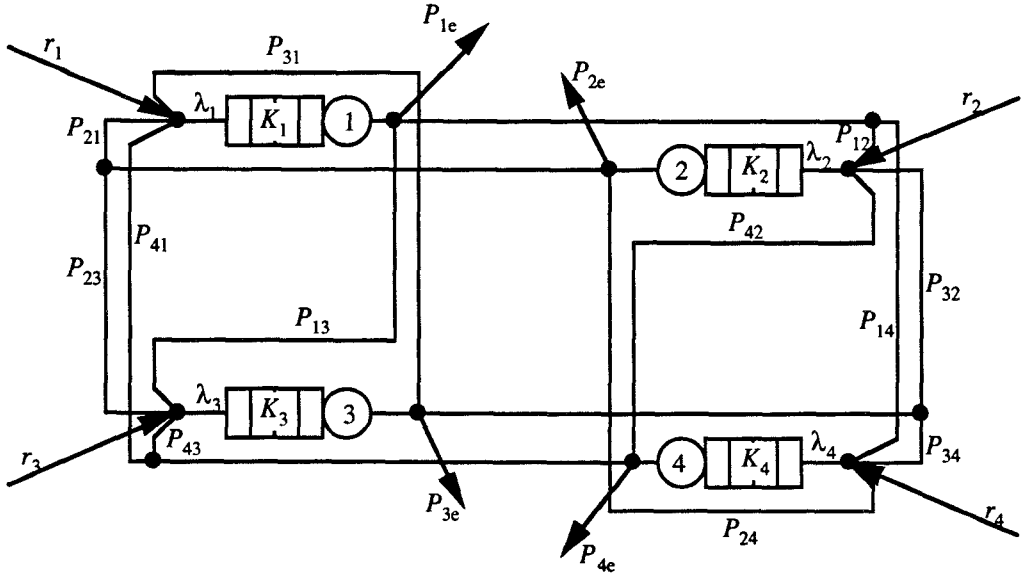


Fig. 8 — A four-queue network

Standard Clock simulation of a network of M/M/1/K queues proceeds in nearly the same manner as for a single queue. The main differences are that the network state must be described by a vector $\boldsymbol{x} = \{x_1, x_2, \ldots, x_J\}$, where x_i = the number of packets occupying queue i, instead of the scalar x used to describe the queue state. Also, the set of events is much larger, and, as a result, the state update function is more complicated.

Although, in essence, all events are either arrivals or departures for both the simple queue and for the network, there are many more event types for the network than there are for the single queue. To simulate the network we must distinguish exogenous arrivals for queue i from those for queue j. Therefore we use the notation a_i to denote an exogenous arrival for queue i. We must also distinguish between departures from queue i bound for queue j and those bound for queue k or exiting the network. Since a departure from queue i that is bound for queue j corresponds to an arrival at queue j, we treat this as a single "transfer" event, denoted t_{ij}. We let t_{ie} indicate that the departure is exiting the network. A "ratio yardstick" showing the set of all events for the network of Fig. 8, along with the probability of their occurrence, is shown in Fig. 9. Because self-loops are not present in Fig. 8, $P_{ii} = 0$ and there are no t_{ii} shown in Fig. 9.

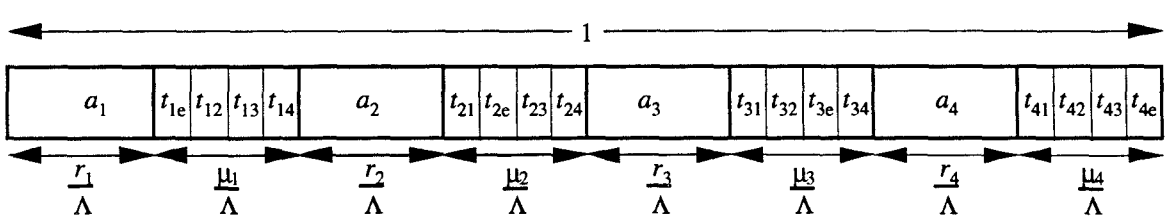


Fig. 9 — Ratio yardstick showing the set of events for the network of Fig. 8

As is implied by Fig. 9, the maximal event rate Λ for our example four-queue network is the sum of the exogenous arrival rates plus the sum of the server rates:

$$\Lambda = \sum_{j=1}^{4} \left(r_j + \mu_j \right).$$

Let us now review the basics of SC simulation, and discuss its application to the problem of interest. Events are generated at the maximal rate Λ with exponential interevent times. The type of event is then determined by generating a random number uniformly distributed between 0 and 1. Conceptually, each event corresponds to a region on the ratio yardstick, where the width of each is proportional to the probability of the corresponding event. Thus each random number falls into one of these regions, resulting in the generation of the corresponding event. In our simulations we have used the alias method to determine the event type. Although it is harder to explain than the ratio yardstick (and we do not do so here, see [24]), it is computationally more efficient than performing a series of comparisons to determine into which region (in this case of 20 such regions) the random number falls. In fact, the computational effort involved in determining the event type by means of the alias method is independent of the number of event types.

Referring to Fig. 9, event types are determined as follows. An event is an exogenous arrival to queue i with probability r_i/Λ. Similarly, it is a departure from queue i with probability μ_i/Λ. As is shown in Fig. 9, this probability quantity is subdivided into the four possible transfer events, i.e., the departure event is a transfer from queue i to queue j with probability

$$P\left(t_{ij}\right) = \begin{cases} \dfrac{P_{ij}\mu_i}{\Lambda} & i \neq j \\ \dfrac{P_{ie}\mu_i}{\Lambda} & j = \mathrm{e} \end{cases}.$$

Notice that the event generation process is independent of the buffer capacities. The same sequence of events used to simulate a network of M/M/1/K queues could also be used to simulate a standard (i.e., infinite buffer) Jackson network with the same topology and routing probabilities. The fact that the queues have finite buffers affects the simulation in only the state update function.

Clearly, with the larger event set required for a network of queues, the state update function must be more complex than the one used for a single queue. At the $(n + 1)^{st}$ event, it is given by

$$x(n+1)=\begin{cases} x(n)+\delta_i & \text{event}=a_i,\ x_i<K_i \\ x(n) & \begin{cases}\text{event}=a_i,\ x_i=K_i \\ \text{event}=t_{ij},\ x_i=0\end{cases} \\ x(n)-\delta_i & \begin{cases}\text{event}=t_{ij},\ x_i>0,\ x_j=K_j \\ \text{event}=t_{ie},\ x_i>0\end{cases} \\ x(n)-\delta_i+\delta_j & \text{event}=t_{ij},\ x_i>0,\ x_j<K_j,\ i\neq j\end{cases},$$

where $x(n) \pm \delta_i = \{x_1(n), \ldots, x_i(n) \pm 1, \ldots, x_J(n)\}$.

5.2 Simulation Results

In Table 2 we show a timing breakdown from a run that simulated the network of Fig. 8 with 100 different parameter settings, i.e., $K_1 = 1, \ldots, 100$; $K_2 = K_3 = K_4 = 100$. The remaining parameters were: $r_i = \mu_i$, $i = 1, \ldots, 4$; $P_{ij} = P_{ie} = 0.25$, $i, j = 1, \ldots, 4$, $i \neq j$. As in Table 1, the time per call for the total event generation category includes two calls to `random`. As expected, the time required per `update` call is larger than that required for the M/M/1/K queue model (it was 4.521μs, see Table 1). However, we see that approximately the same percentage of simulation time is spent actually updating the sample paths in both the network model and the single-queue model.

Table 2 — Timing breakdown for SC simulations of networks queues

	SC simulations; $N = 100$; r_i=0.2, i=1,...4			
function	calls	time (s)	%time	time/call
`update`	10,000,000	59.36	97.4	5.936μs
`get_event`	100,000	0.47	0.8	4.700μs
`random`	200,310	0.69	1.1	3.445μs
`log`	100,000	0.43	0.7	4.300μs
total event generation	100,000	1.59	2.6	15.890μs

These results indicate that the SC method scales well with problem complexity. There are four times the number of queues and 10 times as many events to consider in this network simulation, and the `update` function is more complex than that for the M/M/1/K queue. However, the time per call to the `update` function is only 1/3 longer for the network simulation. A comparison of the time per call to the `get_event`, `random`, and `log` functions (and the resulting total event generation time per call) in Tables 1 and 2 indicates a high level of consistency between our simulations of the two systems. This is not unexpected since `random` and `log` are clearly identical in both systems, and, as mentioned in Section 5.1, the use of the alias method makes the computational effort involved in the `get_event` function independent of the number of event types.

6 SC SIMULATION APPLIED TO CIRCUIT-SWITCHED VOICE NETWORKS

In this section we discuss the application of SC simulation techniques to multihop radio networks that support circuit-switched voice traffic. In Section 6.1 we describe the networking model, which is based on the one studied in [17]. The availability of an exact product-form solution for this model has permitted the validation of our simulation results. In Section 6.2 we describe the SC simulation model, and in Section 6.3 we demonstrate the improvement in efficiency that is achievable by means of SC techniques. We postpone our discussion of the simulation results in terms of network performance to Section 8, where we also discuss the performance of an integrated voice/data network model.

As in [17] we do not address the protocol issues that are associated with call setup, such as the control messages that must be exchanged to request and to verify the availability of network resources needed to support the call. Our focus is on the performance that can be achieved in an idealized system in which the availability of full state information is available at all nodes, with no time delay or overhead cost. Protocol issues will be addressed in the future, and will take advantage of the insights gained through these studies.

6.1 The Voice Network Model

Here we provide a brief description of the voice network model, including the use of "coordinate-convex" policies. A complete description is provided in [17]. Voice-call service is provided by the establishment of a circuit over a predetermined path between the originator of a call (i.e., its source node) and its destination node for the duration of the call. Calls arrive to circuit j according to a Poisson process with rate λ_j, and their lengths are exponentially distributed with parameter μ_j. A vector description of circuit j in terms of the nodes it traverses is given by $c_j = \{c_{j1}, c_{j2}, ..., c_{jN}\}$, where

$$c_{ji} = \begin{cases} 1, & \text{if circuit } j \text{ traverses node } i \\ 0, & \text{otherwise} \end{cases},$$

and N is the number of nodes in the network. The state of the system with J SD pairs, and hence J circuits, is described by the vector $\boldsymbol{x} = \{x_1, x_2, ..., x_J\}$, where x_j is the number of calls currently active on circuit j, each of which is referred to as a call of type j. A central controller makes the decisions on whether or not to accept calls based on perfect knowledge of the number of calls of each type that are currently in progress (i.e., the system state $\boldsymbol{x}$), and hence the set of resources that are available for new calls. The resources (a transceiver at each node in the intended circuit) needed to establish a circuit are acquired simultaneously when the call arrives and are released simultaneously when the call is completed. Calls are blocked when one or more nodes along the path do not have a transceiver available, or when a decision is made not to accept a call. Blocked

calls are assumed to be lost from the system, a mode of operation known as "blocked calls cleared." These assumptions, coupled with the class of admission-control policies discussed below, leads to a mathematically tractable description of system performance.

The number of transceivers at node i, denoted by T_i, represents a fundamental system constraint. No more than T_i calls can simultaneously use the resources at node i, that is

$$\sum_{j=1}^{J} x_j c_{ji} \le T_i, \qquad i = 1,\ldots,N. \tag{5}$$

These equations are termed the "system constraints." The system constraints limit the state space Ω_0 in which $\boldsymbol{x}$ is allowed to take values. We assume that the state space is coordinate convex [26]. The primary characteristic of a coordinate-convex state space is that if $\boldsymbol{x}$ is an admissible state ($\boldsymbol{x} \in \Omega_0$) and $x_j \ge 1$, then $\boldsymbol{x}' = (x_1, x_2, \ldots, x_j-1, \ldots, x_J)$ must also be an admissible state ($\boldsymbol{x}' \in \Omega_0$). This condition implies the very reasonable property that call completions are not blocked, and that call durations are independent of the system state. We refer to a system in which calls are admitted as long as resources are available at all nodes along the path as an "uncontrolled" system.

In some cases, network performance (e.g., blocking probability or throughput) can be improved by blocking calls even though resources are available. We consider admission-control policies that retain the coordinate convexity of the state space. Under such policies, a new call is admitted with probability 1 if the state to be entered is in the admissible region; otherwise, it is blocked with probability 1. A coordinate-convex policy is specified in terms of the set of admissible states, which is defined by a set of linear inequality constraints corresponding to coordinate-convex regions in discrete state space. Thus the control policy is effectively a further restriction of the admissible state space defined by the system constraints (Eq. (5)). Under our assumption of Poisson arrival statistics and exponentially distributed call duration, the use of coordinate-convex policies results in a product-form characterization of the system state, which greatly simplifies the evaluation of system performance, as is discussed below.

For notational purposes, we subdivide the control policy, which we denote as Ω, into a set of circuit "thresholds," and a set of "control constraints." Thresholds restrict the number of calls that will be admitted to the individual circuits, and can be expressed as

$$x_j \le X_j = \text{ threshold on circuit } j, \qquad j = 1,\ldots,J.$$

The control constraints are restrictions on the sums of the number of calls of various types, i.e.,

$$\sum_{j \in I} x_j \le Y_I$$

for various sets I, where I is a subset of the set of all call types. Thus, the control policy is given by $\Omega = \{X_1, \ldots, X_J, Y_1, \ldots\}$. Now the problem is the determination of the optimal admission-control policy Ω^*, i.e., the values of the X_j and the Y_I that yield the minimum value of blocking probability.

In [17, 27, 28], we developed an analytical model to examine the effects of administering different admission-control policies in the five-circuit network shown in Fig. 10. With $T_i = T$, $i = 1, \ldots, 10$, the system constraints are for this network are:

$$x_1 + x_2 \le T \qquad x_1 + x_3 + x_5 \le T \qquad x_1 + x_4 + x_5 \le T \qquad x_j \le T.$$

We consider administering control by adjusting circuit thresholds and control constraints.

$x_j \le X_j$ (thresholds)

$x_1 + x_3 \le Y_1$ (control constraints from node 5)

$x_1 + x_4 \le Y_2$ (control constraints from node 7)

$x_1 + x_5 \le Y_3$ (control constraints from node 5)

$x_3 + x_5 \le Y_4$ (control constraints from node 5)

$x_4 + x_5 \le Y_5$ (control constraints from node 7)

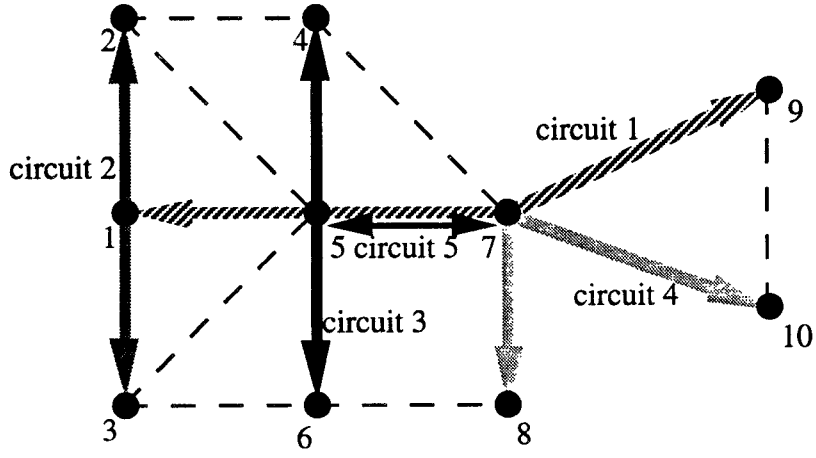


Fig. 10 — A circuit-switched voice network

The use of this form of control policies assures us that the state space is coordinate convex [26, 29, 30]. With our assumptions of Poisson arrivals (at rate λ_j on circuit j) and exponential call duration (with expected length $1/\mu_j$ on circuit j), the use of coordinate-convex policies results in a product-form characterization of the voice state that is given by

$$\pi(\boldsymbol{x}) = \pi(0)\prod_{j=1}^{J}\frac{\rho_j^{x_j}}{x_j!},$$

where $\rho_j = \lambda_j/\mu_j$, and $\pi(0)$ is the normalization constant given by

$$\pi(0) = \left\{ \sum_{x \in \Omega} \prod_{j=1}^{J} \frac{\rho_j^{x_j}}{x_j!} \right\}^{-1}.$$

For any coordinate-convex policy, it is straightforward (though time consuming) to evaluate $\pi(0)$, which in turn permits the evaluation of performance measures such as throughput and blocking probability. Blocking probability $P_b(\Omega)$ is the ratio of the expected number of blocked calls per unit time to the expected total number of call arrivals per unit time:

$$P_b(\Omega) = \frac{\sum_{j=1}^{J} \rho_j P_{bj}(\Omega)}{\sum_{j=1}^{J} \rho_j} = \frac{\sum_{x \in \Omega} \sum_{j=1}^{J} 1\left(\left(x_j + 1\right) \notin \Omega\right) \rho_j \pi(x)}{\sum_{j=1}^{J} \rho_j},$$

where $P_{bj}(\Omega)$ is the fraction of type-j calls that are blocked, and $1(\cdot)$ is the indicator function, which is 1 if the argument is true and 0 otherwise.

6.2 A Standard Clock Model for Circuit-Switched Voice Networks

We have used SC techniques to simulate the performance of the circuit-switched network of Fig. 10 under a number of different admission-control policies. The availability of the exact solutions obtained in [17, 27, 28] has permitted us to validate the results of our simulations. In this subsection we describe the SC model we have used for this example.

The discrete parameters of interest in this case are the circuit thresholds X_j and the Y_I. The events that must be generated are arrivals and departures. An arrival to circuit j is denoted a_j, and occurs at a rate of λ_j. The departure rate for each active call on circuit j is μ_j; thus, if there are x_j active calls on circuit j the departure rate for calls of type j is $x_j\mu_j$. This situation is translated to events in the simulation as follows. Because there may be up to X_j active calls on circuit j, we must consider X_j different departure events d_{jn} (n = 1, ..., X_j) for circuit j, namely,

$d_{j1} \Rightarrow$ feasible departure from circuit j if $x_j \geq 1$, otherwise fictitious event;

$d_{j2} \Rightarrow$ feasible departure from circuit j if $x_j \geq 2$, otherwise fictitious event;

and in general,

$d_{jn} \Rightarrow$ feasible departure from circuit j if $x_j \geq n$, otherwise fictitious event.

Thus, the maximal rate of this system is

$$\Lambda = \sum_{j=1}^{J} \left(\lambda_j + X_j \mu_j\right).$$

In our SC simulation, events are generated at the maximal rate Λ with exponential interevent times. The type of event is then determined by generating a random number uniformly distributed between 0 and 1. The upper part of Fig. 11 shows the "ratio yardstick" for the maximal system for an example in which the maximum threshold on each circuit is three; also, $\lambda_j = \mu_j = 1$, which implies that all events are equally likely.[5] As in our discussion of the queueing network in Section 5.1, each event corresponds to a region on the ratio yardstick, where the width of each is proportional to the probability of the corresponding event. We again use the alias method to determine the event type because of its improved efficiency as compared to the use of a series of comparison operations.

All arrival events in this system are real. However, departure events are fictitious if an insufficient number of calls of that type are currently active, as discussed above. For example, referring again to the upper part of Fig. 11, an event of type d_{13} will be fictitious (and hence ignored, although time will be updated) if there are less than three calls of type 1 currently active.

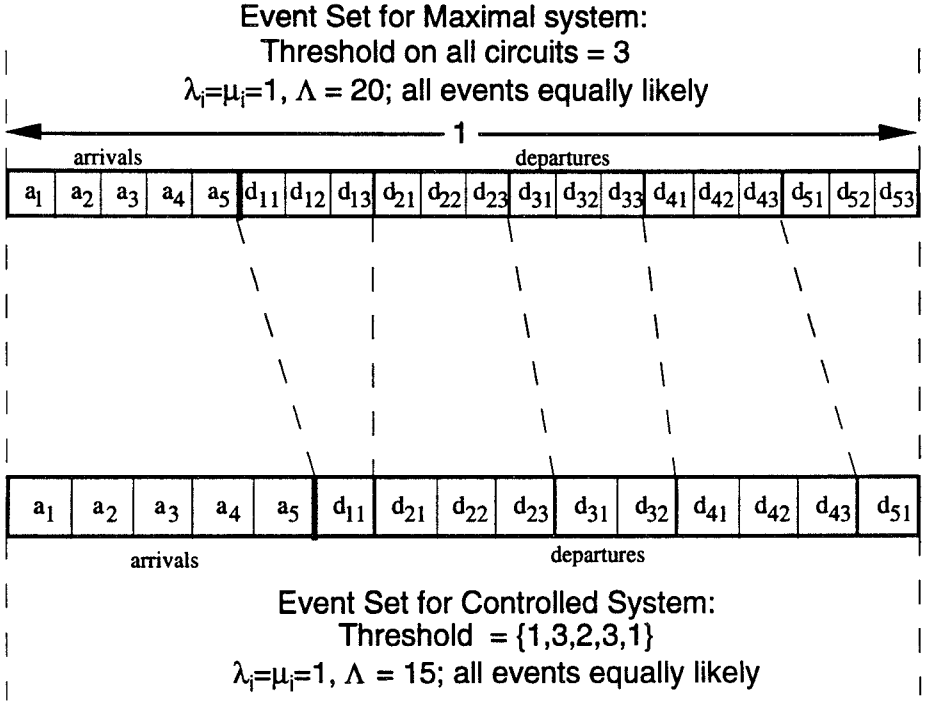


Fig. 11 — Ratio yardsticks showing event sets for the circuit-switched network

To reduce the number of fictitious events (and thus improve the efficiency of the simulations), we have considered the use of custom ratio yardsticks (implemented by means of custom alias tables, i.e., alias tables constructed to conform with the particular parameters of each sample path). The lower part of Fig. 11 shows a custom ratio yardstick for an example in

[5] If the events were not equally likely, the boxes would have different widths, which are proportional to the corresponding probabilities.

which the circuit thresholds have been lowered to {1,3,2,3,1}. A number of events have been eliminated because they will always be fictitious; e.g., with the threshold on circuit 1 reduced to 1 in the controlled system, the departure events d_{12} and d_{13} will always be fictitious. Clearly, by eliminating them from the alias tables we can increase the percentage of real events. The remaining events are again equally likely, although there are only 15 of them instead of 20.

Despite the potential advantages of custom alias tables, we have not found their use to be beneficial in our SC simulations. Clearly, they are not necessary because the alias tables constructed for the maximal (uncontrolled) system can be used for all sample paths. Such maximal alias tables provide all feasible events for every sample path, since the thresholds can only be reduced from those of the uncontrolled system; this is, in fact, the basic principle of the SC approach. In this case, the probability of the occurrence of each event is determined from the maximal system, and the events are appropriately "thinned" for the different sample paths by applying their individual constraints. The obvious disadvantage of this approach, namely increasing the number of fictitious events, is usually overshadowed by the following advantages:

(1) Reduced memory requirements;

(2) Each sample path sees the same event sequence, hence

(3) All sample paths require the same number of events to see the same number of arrivals.

(4) One call to `get_event` per event—custom alias tables require one call to `get_event` per event per sample path.

However, there certainly may be some applications in which the use of custom alias tables is beneficial, e.g., in cases where the percentage of events that are fictitious is high.

6.3 Simulation Timing Results

We have performed both BF and SC simulations of this network model to obtain timing results. As in our studies of the M/M/1/K queue model, our focus in this section is on the increase in simulation efficiency made possible by the use of the SC method rather than the actual network performance. However, we have validated our simulations by comparing the network performance statistics (blocking probability) from our simulations with the exact values given in [27], and in Section 8 we discuss these performance results. In these simulations, N_{sp} different sample paths, each having a different control policy, are generated until they have received a given number of events. In the SC simulations we used a single alias table for all sample paths, but custom alias tables were used in the BF simulations to reduce the number of fictitious events.

Figure 12 compares the total time required to simulate N_{sp} sample paths using BF and SC methods as N_{sp} varies from 100 to 5,000 for BF simulations, and from 100 to 15,000 for SC simulations. The timing results shown in Fig. 12 are based on averages taken over 10 runs using different random number generator seeds. The simulation of each sample path was terminated when it had received 1,000 voice arrival events. Note that the total simulation time in both cases is roughly linear in N_{sp}. Here again, we see that a significant time savings is achieved by using SC instead of BF simulation. For N_{sp} greater than 2,000, the SC method is approximately 2.5 times faster than the BF approach. For long simulations this speedup can save hours of computer time.

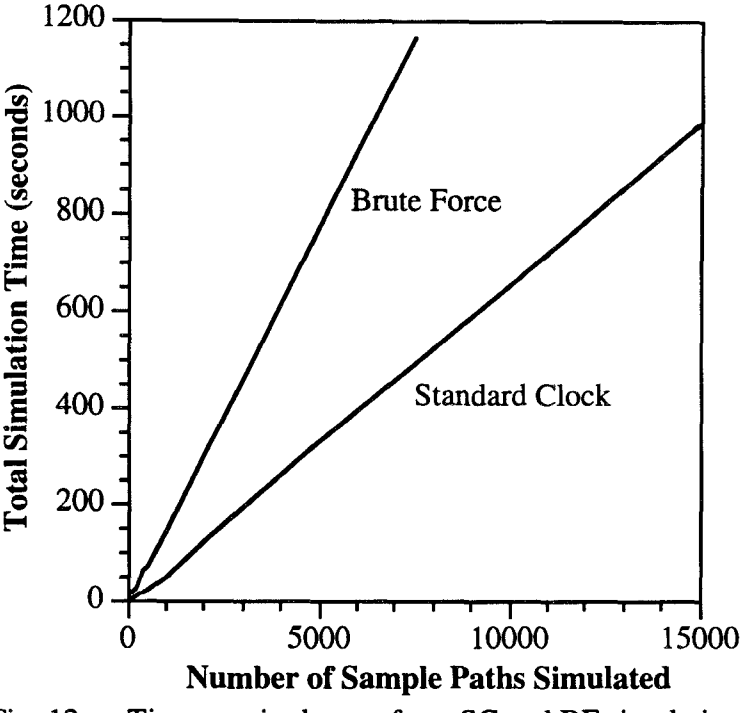


Fig. 12 — Time required to perform SC and BF simulations of the circuit-switched voice network

The primary source of the time savings shown in Fig. 12 is the reduction in the number of events that must be generated in a SC simulation, as can be seen in Table 3. This table presents timing breakdowns for SC and BF simulations. The runs used to generate this data simulated 100 sample paths, and were terminated when each sample path had received 10,000 arrivals. Notice that for all four functions the time per function call is approximately the same as in Table 1, the M/M/1/K timing breakdown.

Despite the 17.5-fold increase in the size of the event set for the circuit-switched voice model (there are 35 events in the voice model, 2 in the M/M/1/K model), the time per call for the functions `update` and `get_event` is approximately the same as that in the M/M/1/K simulations. Although there are many more lines of code and many more constraints in the

voice-model `update` function, the operations actually performed are virtually the same as those in the M/M/1/K model. In both models, a switch statement based on the event is used to perform the appropriate set of comparisons and accordingly update the state and certain statistics. The `get_event` function is exactly the same in both models; the only difference is the size of the alias tables that are accessed within the function. Based on these results, we have found that the time per call to the `get_event` and `update` functions is relatively insensitive to the problem size and complexity. Thus, the SC approach should scale well, and may actually provide more performance improvement with more-complex problems.

Table 3—Timing breakdown for BF and SC simulations of the circuit-switched voice network

	BF				SC			
function	calls	time (s)	%time	time/call	calls	time (s)	%time	time/call
`update`	4682117	27.22	25.7	5.8136μs	6892000	24.25	94.5	3.5186μs
`get_event`	4682117	24.36	23.0	5.2028μs	68920	0.45	1.8	6.5293μs
`random`	9364234	33.03	31.2	3.5273μs	137840	0.65	2.5	4.7156μs
`log`	4682117	21.40	20.1	4.5706μs	68920	0.30	1.2	4.3529μs
total event generation	4682117	78.79	74.3	16.828μs	68920	1.40	5.5	20.3134μs

7 SC SIMULATION OF INTEGRATED VOICE/DATA NETWORKS

We have noted that the SC approach is applicable only when the interevent times in the system are exponentially distributed. However, Ho et al. [14] have shown that certain deterministic events can also be incorporated into the simulation, which they refer to as "standard control clock (SC^2)" events. In this section we show how integrated networks with fixed-length (single-packet) data traffic can be modeled by using the SC approach. Also, we shift our focus from the study of the efficiency of the SC method to the performance evaluation of the system being studied, with the ultimate goal being the optimization of system performance. Performance results are presented in Section 8.

7.1 The Integrated Network Model

It is straightforward to extend our simulation to the case of an integrated network such as that shown in Fig. 13, in which packet-switched data with fixed-length packets is supported along with circuit-switched voice of exponential call duration. The voice traffic model is identical to the one discussed in the previous section, i.e., Poisson arrivals at rate λ_j^V and exponential service with parameter μ_j^V on circuit j. Data traffic consists of single fixed-length packets with Poisson arrival statistics. Note that both voice and data arrival events, as well as

voice departure events, are stochastic events. However, because the data packets are of fixed length, the packet service rate and hence the data-packet departure events are deterministic (given the voice state and data-packet arrival times).

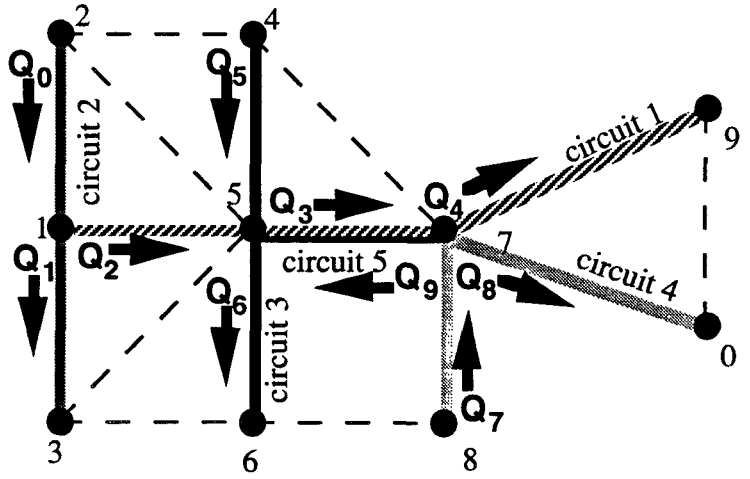


Fig. 13 — An example voice and data network

At each node a separate queue is formed for packets intended for each of its neighbors; e.g., in Fig. 13 the queues are denoted as Q_0, Q_1, ..., Q_9. In particular, the arrival process at queue k is Poisson at rate λ_k^d. The transceivers not being used at that time to support voice traffic are available to support data traffic. Data packets are handled in our simulation model as follows. Whenever one or more transceivers at a node are idle, the data queues at the node are examined. If any are nonempty, and if a transceiver is also available at the destination node, the packet is transmitted; each data-packet transmission occupies one transceiver at both source and destination nodes for the duration of the packet. We have also investigated an alternative model in which the availability of a transceiver is not required at the receiving node. This latter case implicitly assumes that receivers are abundant, and that only transmitters are a limited resource. We refer to these two models as "receivers verified" and "receivers assumed" respectively.

These data-traffic models neglect the protocol issues associated with the coordination of transmitting and receiving platforms; as noted earlier, our focus here is on the performance that can be achieved in an idealized system in which full state information is available at all nodes, with no time delay or overhead cost. However, we would like to address briefly how data traffic might be handled in a practical system. One approach would be to establish a link-activation schedule that pairs neighboring nodes on a time-division basis [31, 32]. Such a schedule could be established by initially assuming that all time slots at all transceivers are available for data traffic. Then, as each voice call is admitted, it would remove one transceiver from the pool available for data transmission at each node along the multihop path; when calls are completed,

the transceivers would become available for data again. The study of such protocols for integrated channel access is a topic for future study.

In our simulations we have used unlimited capacity buffers at each queue. However, finite capacity buffers can easily be included by using the M/M/1/K model developed in Section 3. In the example shown, we assume that data uses only those links that are being used for one or more voice circuits, although that can be generalized trivially to include all potential links in the network. The system state can then be described by the vector pair $\{\boldsymbol{x}, \boldsymbol{d}\}$, where $\boldsymbol{x}$ represents the number of voice calls of each type as described earlier, and $\boldsymbol{d}$ represents the number of data packets in each queue.

Voice traffic has priority over data traffic in the following sense. The decision to accept a voice call depends only on the number of voice calls (of each type) currently in progress in the system, as in the voice-only system of Section 6; the size of data queues does not affect this decision. Data packets are permitted to use any transceivers that are not occupied by voice, and are queued when resources are not immediately available. If a voice call arrives when data traffic is using one or more of the transceivers needed to support the call, the data packet is permitted to complete its transmission, after which the voice call begins. Since data packets are generally of much smaller length than voice calls, this should not interfere significantly with the assumption of exponential call lengths.[6]

The simplest case to consider is that of single-hop data traffic, i.e., all traffic generated at a node is to be delivered to one of its immediate neighbors (no relaying is required). Certainly, the case of multihop traffic (which requires relaying by one or more intermediate nodes) is of greater interest for practical applications. However, no exact models are available that characterize delay over multihop paths. Multihop systems may be modeled approximately by assuming a Poisson arrival stream of appropriate rate at each queue along the multihop path,[7] thus implicitly assuming that the queues at each node are independent; this is similar to Kleinrock's well-known independence assumption [33]. The expected end-to-end delay would then be the sum of the expected delays at each hop along the path. Of course, this model is not exact, e.g., because the arrival processes at intermediate nodes are not Poisson and are not independent. Future studies will investigate the accuracy of the use of this independence

---

[6] Actually, as noted earlier, the performance of circuit-switched networks with Poisson arrival statistics depends primarily on the expected call length, and is quite insensitive to the call-length distribution. Alternatively, it could have been assumed that the data packet is preempted, necessitating its retransmission when the channel again becomes available.

[7] For example, let us consider a source-destination (SD) pair that uses a two-hop path consisting of links i and j. We assume that data packets for this SD pair arrive at rate λ according to a Poisson arrival process. The approximation consists of assuming that the traffic associated with this SD pair adds independent Poisson input streams of rate λ to the queues associated with links i and j.

assumption for multihop networks. However, for simplicity, in this report we assume that data traffic is single hop.

7.1.1 Performance Measures

As in Section 6, we wish to evaluate the effect of administering a voice admission-control policy. Now, however, we can evaluate the effects of the control policy on performance metrics that incorporate aspects of both voice and data performance.

Residual Data Capacity

In [17] we introduced data metrics that are based solely on knowledge of the voice statistics. From the steady-state distribution of the voice-call process, we can calculate the average number of active calls of each type (i.e., the expected voice state). At any individual node the "residual data capacity" is defined to be the expected number of transceivers available for data, i.e., not being used for voice traffic. This definition is easily extended to a network-oriented performance metric. Since data traffic is free to use a resource (transceiver) whenever it is not occupied by voice traffic, the residual data capacity of an integrated network with given voice-utilization rates is an indicator of potential network performance with respect to data. We have considered two different measures of residual network data capacity. The *mean data capacity*, denoted C_m, is the mean of the residual data capacity at each node, where the average is taken over all nodes in the network. The *bottleneck data capacity*, denoted C_b, is the minimum residual data capacity at any individual node, where the search is performed over all network nodes. These metrics allow us to rank admission-control policies based on a combined network performance metric

$$\Theta_*(\Omega,\alpha) = \alpha C_* + (1-\alpha)\Gamma(\Omega),$$

where $\Gamma(\Omega)$ is the voice throughput, $\alpha \in (0, 1)$ is the weighting factor, which is chosen to reflect the relative importance of the voice and data performance metrics, and the subscript on Θ and C indicates which data metric, i.e., bottleneck data capacity C_b or mean data capacity C_m, is being used.

When data traffic levels are not specified a priori, the residual data capacity is a reasonable means by which to incorporate the impact of data traffic on overall system performance. This metric provides an indication of how much data traffic can be supported by a network, given its resources (i.e., number of transceivers at each node) and the voice traffic that is to be supported. However, this indication is very imprecise because, under the receivers-verified model, a node will be able to use all of its residual capacity for data only if a sufficient number of transceivers are available at its neighbors; in a radio network a transceiver (which may be used for either transmission or reception, but not both simultaneously) must be available at

both the transmitting and receiving nodes.[8] Thus the "usable residual capacity," would typically be lower than computed by considering the nodes individually as we do in the receivers-assumed model. To evaluate the usable residual capacity accurately in our simulation model that requires a transceiver at the receiving node (i.e., the receivers-verified model), it is necessary to keep track of the number of transceivers in use at all network nodes; thus a packet can be transmitted only when a receiver is available at the destination node.

Nevertheless, our simulation results suggest that the residual data capacity does indeed provide a reasonable characterization of system performance. Of the two performance measures, C_m may be of more importance if the primary concern with data traffic is the average data throughput that is supported by the network. However, C_b may be more appropriate if delay is of primary importance, since an infinite delay will result if the offered data rate over any link is greater than the residual data capacity available over that link.

Delay

When data traffic levels are specified, appropriate data performance metrics include the throughput actually supported by the network as well as the average delay experienced by data packets. As noted above, the transmission of a data packet requires the availability of a transceiver at both the transmitting and receiving nodes. Thus the evaluation of data-packet delay would depend on the specific protocol used to activate links for data transmission. We have used a very simplified model for the availability of data links in which we address only the number of transceivers available at each transmitting node, while neglecting the need to have one available at each receiving node. In doing so, we model the data process at node i as an M/D/C_i queue,[9] where C_i is the residual data capacity at node i.

It is well known that the expected delay of an M/D/1 queueing system (not including service time) is:

$$D = \frac{\rho}{2\mu(1-\rho)},$$

where $\rho = \lambda/\mu$ is the average load. There is no simple expression of this type for the M/D/c queue (a numerical procedure to determine its probability mass function and waiting time is described in [34]); however, its performance can be approximated roughly by that of the M/D/1 queue, where ρ is appropriately redefined as λ/μ' with $\mu' = c\mu$. To reflect this type of behavior, we have used the following approximate expression to estimate data-packet delay at node i:

---

[8] In a wireline network there is no such problem of matching transmitting and receiving nodes, and residual capacity is a more well-defined notion.

[9] The notation M/D/c refers to a queueing system with Poisson (Markovian) arrivals, deterministic service time, and c servers.

$$\tilde{D}_i(\Omega) = \frac{\rho_i}{2C_i(1-\rho_i)},$$

where

$$\rho_i = \frac{\sum_{k\in i}\lambda_k^d}{C_i},$$

and μ is set equal to 1 without loss of generality. Here, $\sum_{k\in i}\lambda_k^d$ is the total data arrival rate at node i, where the sum is taken over all queues k that reside at note i, C_i = residual data capacity at node i, i.e.,

$$C_i = T_i - \sum_{j=1}^{J} c_{ji}\bar{x}_j,$$

where T_i is the number of transceivers at node i,

$$c_{ji} = \begin{cases} 1, \text{ node } i \in \text{ circuit } j \\ 0, \text{ otherwise} \end{cases},$$

and

$$\bar{x}_j = \sum_{n=1}^{X_j} nP(x_j = n) = \sum_{n=1}^{X_j}\left\{ n \sum_{\{x:x_j=n\}} \pi(x) \right\}$$

is the expected number of active calls on circuit j.

The delay metric we have used is the average nodal delay taken over all nodes

$$\tilde{E}\{D(\Omega)\} = \sum_{\forall \text{ nodes } i}\left(\tilde{D}_i(\Omega) \frac{\sum_{k\in i}\lambda_k^d}{\sum_{\forall \text{ queues } k}\lambda_k^d} \right).$$

This delay estimate can be used in a network metric that uses a convex combination of the voice blocking probability ($P_b(\Omega)$) and the expected delay $\tilde{E}\{D(\Omega)\}$:

$$\Theta(\Omega,\alpha) = \alpha P_b(\Omega) + (1-\alpha)\tilde{E}\{D(\Omega)\}.$$

In addition to the fact that the above expression for delay ignores the need to pair transceivers at the transmitting and receiving nodes, it also ignores other critical aspects of network operation. Most significant is the implicit assumption that the number of data packets that are serviced per unit time at node i is constant at C_i (the fact that C_i is in general non-integral is a minor point here), whereas the number of servers available in the real system varies, based on the voice state. An accurate estimate of delay would have to take into account not only the expected voice state (which determines residual capacity) as we do in our model, but also the fraction of time spent in each voice state as well as the statistics of the time duration spent in

each state.[10] Thus, not surprisingly, the delay estimate based on this model is not accurate. However, we show in Section 8.2 that the delay estimate computed in this manner provides a remarkably accurate indication of the relative performance of a large number of different policies, as demonstrated by SC simulations.

7.2 SC Simulation of an Integrated Network

As mentioned earlier, we must deal with both stochastic and deterministic events in the integrated network model. The stochastic events, which are listed in Table 4, include voice arrivals and departures, and data-packet arrivals. As noted earlier, data-packet departure events are deterministic, and can be inferred from the stochastic events in the system. The stochastic events are generated in the usual way, i.e., by drawing two random numbers (one is used to determine the interevent time, one is used to determine the event type) and using the alias method, at a maximal event rate of

$$\Lambda = \sum_{j=1}^{J} (\lambda_j^V + X_j \mu_j^V) + \sum \lambda_k^d,$$

where X_j is the threshold on calls of type j. The system state is updated whenever one of the stochastic events occurs.

Deterministic events are scheduled and processed, as needed, when the state is updated as a result of the occurrence of a stochastic event. Deterministic event (data-packet departure) scheduling occurs when a data-packet transmission begins, i.e., when:

- A data packet arrives at a link with available transceivers (i.e., a transceiver is available at both the queue's source and destination nodes).
- A data-packet departure provides a link with available transceivers for queued data (provided that the channel is not seized by a voice call).
- A voice departure provides link(s) with available transceivers for queued data.

When a deterministic event is scheduled, it is time-stamped and placed in a deterministic events queue. At the occurrence of each stochastic event, the deterministic events queue is checked for eligible events (there may be more than one), which are processed before the stochastic event.

[10] The fraction of time spent in each state is easily determined from the product-form solution. However, the time spent in each state is harder to estimate because it is not reflected in the product-form characterization. We have developed accurate approximate models to characterize the time-varying behavior of the voice state. These models will be discussed in a future report.

Table 4 — Stochastic events

event	parameter	description
1	λ_1^V	voice arrival on circuit 1
		::
5	λ_5^V	voice arrival on circuit 5
6	μ_1^V	voice departure from circuit 1, $x_1 \geq 1$
7	μ_1^V	voice departure from circuit 1, $x_1 \geq 2$
8	μ_1^V	voice departure from circuit 1, $x_1 \geq 3$
9	μ_1^V	voice departure from circuit 1, $x_1 \geq 4$
10	μ_1^V	voice departure from circuit 1, $x_1 \geq 5$
11	μ_1^V	voice departure from circuit 1, $x_1 \geq 6$
12	μ_2^V	voice departure from circuit 2, $x_2 \geq 1$
		::
35	μ_5^V	voice departure from circuit 5, $x_5 \geq 6$
36	λ_0^d	data arrival to queue 0
36+k	λ_k^d	data arrival to queue k

The notion of an "available" transceiver means that at least one transceiver at the node is not committed (at that time instant) to supporting either voice or data traffic, and hence is available to support the transmission of the packet in question.[11] Unlike the simplified analytical model of Section 7.1, the simulation model can check for the availability of the necessary transceivers at the receiving nodes. Since time is continuous in the simulation, there is no possibility of two events happening simultaneously, and thus no ambiguity as to which data packet would be transmitted. As noted earlier, the acceptance of a voice call depends only on the voice-traffic state. Voice calls cannot be blocked by data packets that are using transceivers needed for the call; we simply assume that the data packets complete their transmission after which the voice call claims all needed resources.

Evaluation of Delay via Little's Formula

The "difficult" way to evaluate packet delay is to time stamp each packet upon its arrival and to evaluate its total time in queue when it is transmitted. However, the delay-evaluation process can be simplified greatly by taking advantage of Little's result, which is expressed simply as follows:

[11] As noted in Section 6, our model does not address "protocol" issues such as the transmission of control packets to coordinate the transmitting and receiving nodes. We assume an idealized model in which the nodes pair themselves as needed, without incurring any time delay or overhead.

$$\overline{N} = \lambda T,$$

where $\overline{N}$ is the average number of customers in the queueing system, λ is the average arrival rate (there are no restrictions on type of arrival process, e.g., it does not have to be Poisson), and T is the average time each customer spends in the system (see e.g., [18]). Thus knowledge of average queue size $\overline{N}$ at a node is sufficient to determine the average packet delay. Since we assume a Poisson arrival process for the data-packet stream at each queue, we can take advantage of the fact that Poisson arrivals see time averages (the so-called PASTA property), as we did in Section 4.1. As a result of this property, it is sufficient to evaluate the average queue size at arrival instants to determine the average queue size over all time for use with Little's formula. Since the queue sizes at arrival instants are already monitored as part of the simulation process, it is straightforward to obtain the average queue size, and hence average queueing delay. Since Little's formula is an asymptotically exact result, the accuracy of the delay estimate is as good as that of the average queue size, which depends on the length of simulation run.

8 SIMULATION RESULTS

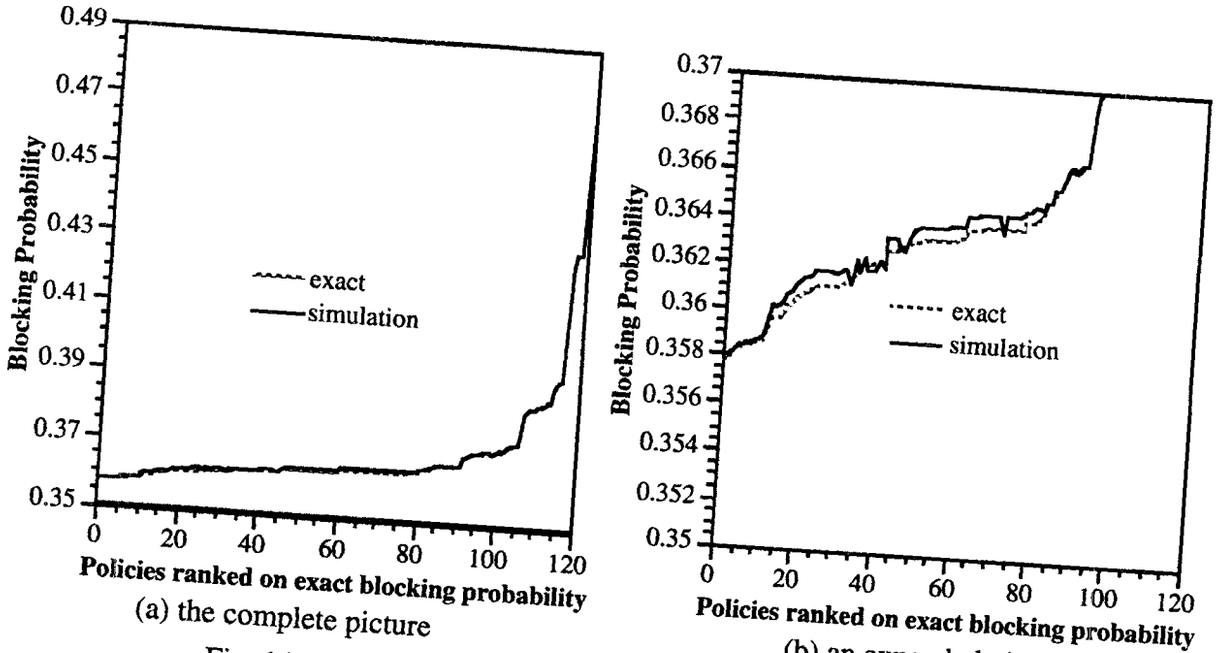
We have performed SC simulations of the integrated network shown in Fig. 13, using the model discussed in Sections 6 and 7. Recall from Section 6 that a policy for this network can be written as $\Omega = \{X_1, \ldots, X_5, Y_1, \ldots, Y_5\}$. We have simultaneously evaluated the 120 different control policies $\{X_1, 6, 6, 6, X_5, 8, 8, Y_3, 8, 8\}$, where $X_1, X_5 = 0,\ldots 6$; $Y_3 = 0,\ldots,8$; $Y_3 \leq X_1 + X_5$; $X_1 \leq Y_3$; and $X_5 \leq Y_3$. We have used various values of $\lambda_j^V = \lambda^V$, $\mu_j^V = \mu^V$, $j = 1, \ldots, 5$, all of which are subject to $\lambda^V/\mu^V = \rho^V = 4.0$. For data, we have set $\lambda_i^d = \lambda^d = 5$, $\mu_i^d = \mu^d = 10$, to yield $\rho_i^d = \rho^d = 0.5$, $i = 0, \ldots, 9$.

In Section 8.1 we discuss the voice-call blocking probability from the perspective of the accuracy of simulated results (as compared to the exact product-form model), as well as the ability of the SC simulation process to find good voice-call admission-control policies. Note that the results for blocking probability are independent of the data traffic, since voice traffic has priority over data. In Section 8.2 we discuss the average data-packet delay (i.e., the average of the expected delay at the ten queues) when the network uses different voice admission-control policies.

8.1 Voice-Call Blocking Probability

In Fig. 14 (a) we show the exact and simulated voice-call blocking probability associated with the 120 control policies. The exact results are determined numerically from the product-form solution, and simulated results are based on a run of one million voice arrival events. Figure 14 (b) is an expanded view of the same curve, which permits a more-detailed comparison

of exact and simulated results. As noted earlier, the results for blocking probability are independent of data traffic parameters; in fact, they do not depend on the individual values of λ_j^V and μ_j^V, but only on the ratios $\rho_j^V = \lambda_j^V / \mu_j^V$. The horizontal axis is simply the ordering from the best (minimum blocking probability based on the exact model) policy to the worst; thus blocking probability is a monotonically nondecreasing function of the horizontal axis. It is significant that there is little sensitivity to the policy that is used when blocking probability is the only performance measure of interest. For example, as the policies are examined from the best to the 80th out of 120, the blocking probability increases by only a small amount, i.e., from 0.358 to 0.365.



(a) the complete picture

(b) an expanded view

Fig. 14 — Probability of blocking across the range of policies; $\rho^V = 4$, simulation duration = 10^6 voice arrival events

It is apparent from the closeness of the curves in Figs. 14 (a) and 14 (b) that the simulation results are extremely accurate. Figure 15 examines this question more closely by plotting the simulation error (defined as %error = 100*abs((exact–simulation)/exact)) as a function of policy ranking. The simulation error is never more than 0.2%.

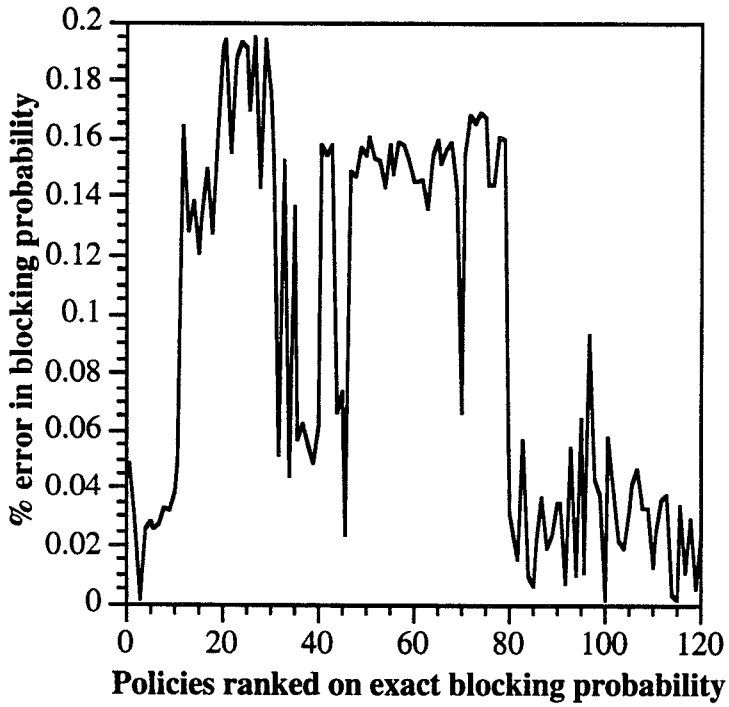


Fig. 15 — Percentage difference between the exact blocking probability and the value obtained through simulation

Ordinal Optimization

In many practical problems involving DEDS, particularly where analytical solutions are not available (as is frequently the case), simulation is the only approach available for the evaluation of system performance. Of course, the well-known drawbacks of simulation are its cost and time expenditure, particularly when one is searching a large search space for the optimal policy. Recently, Ho et al. [35] proposed the use of "ordinal optimization" as an approach to find "good," although not necessarily optimal, solutions to problems involving DEDS. Here, *ordinal* refers to the ranking of policies from best to worst, in contrast to *cardinal* optimization, which evaluates the performance resulting from each policy.

The motivation underlying ordinal optimization is that it is often not worth the degree of effort that is needed to find the optimal solution (or control policy) when a suboptimal solution (that may be found quite easily) will provide sufficiently good performance. For example, to find an optimal policy using standard simulation techniques requires that the simulation be carried out for each policy for a sufficiently long duration to ensure that accurate results are obtained. In contrast, ordinal optimization relies on the use of relatively short simulation runs to obtain a rough ranking of a number of policies; typically, many of the policies that perform well in these short simulations will also do well over the long run. Thus although the measured performance may not be very accurate, the ranking of policies is relatively immune to the effects of "estimation noise."

Typically, a set of policies is selected at random from a much larger population, which is too large to examine exhaustively. The principles of order statistics [36] suggest that if a sufficient number of such random policies are examined, some good ones will be encountered. The more-promising policies can then be examined in greater detail. One way to do so is simply to perform a longer simulation run for each of them. A potentially better approach is to start a search from each of the best policies, possibly in conjunction with adaptive techniques such as genetic algorithms.

Ordinal Ranking of Policies in Terms of Voice-Call Blocking Probability

In Fig. 16 we compare ordinal rankings of the voice blocking probability obtained from two SC simulations to the exact rankings. For these studies we used $\lambda^V = 4$, and $\mu^V = 1$. The two SC simulations differed only in their duration. The short one was terminated after processing 10,000 voice-call arrivals; the longer simulation processed 10^6 voice-call arrivals. The ordinal ranking of the simulation results shows remarkable agreement with the exact ordinal ranking, especially for the longer simulation (ideally the curve would be a straight line with unit slope). Thus despite the insensitivity of blocking probability to the policy used, as discussed earlier, the SC simulation has ranked the 120 policies remarkably well, from best to worst.

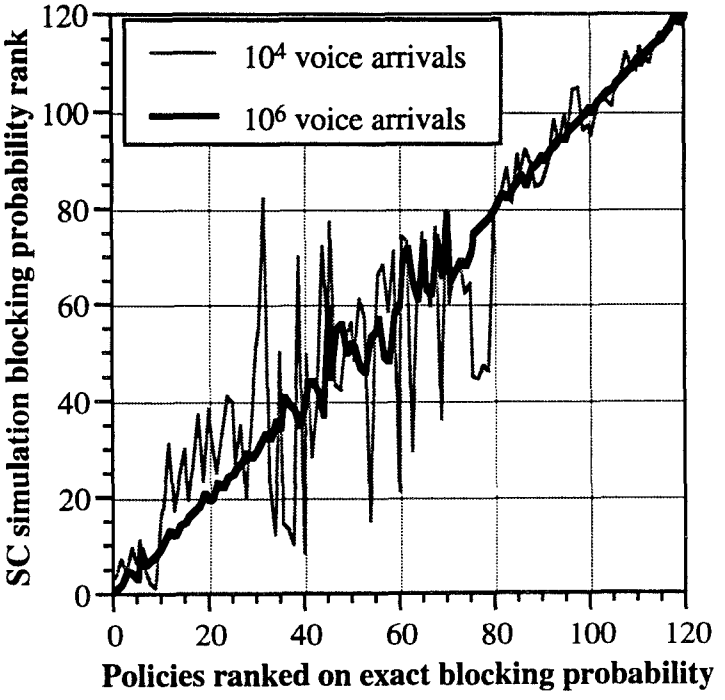
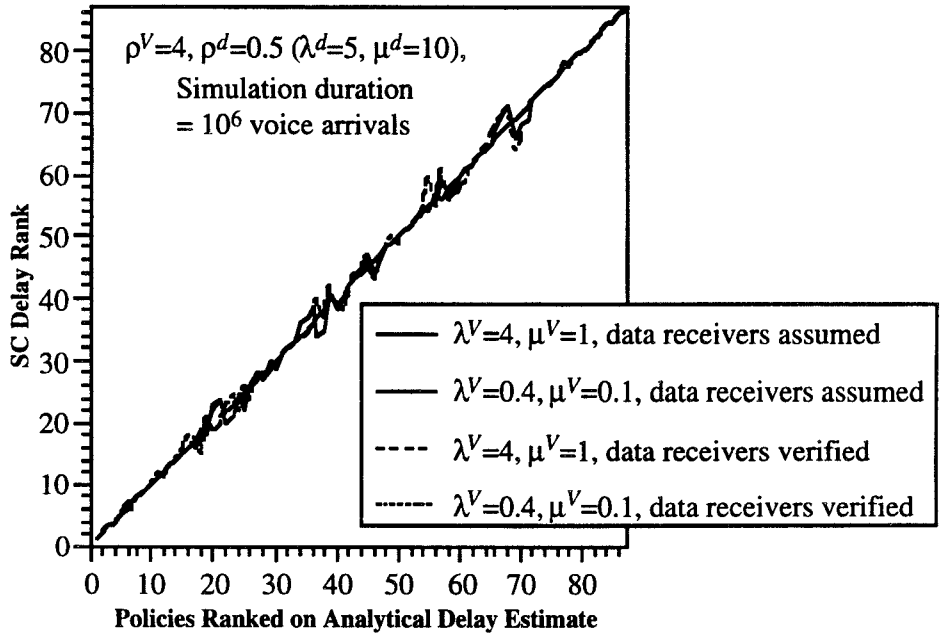


Fig. 16 — Blocking probability found in SC simulations compared to the exact value, $\rho^V = 4$

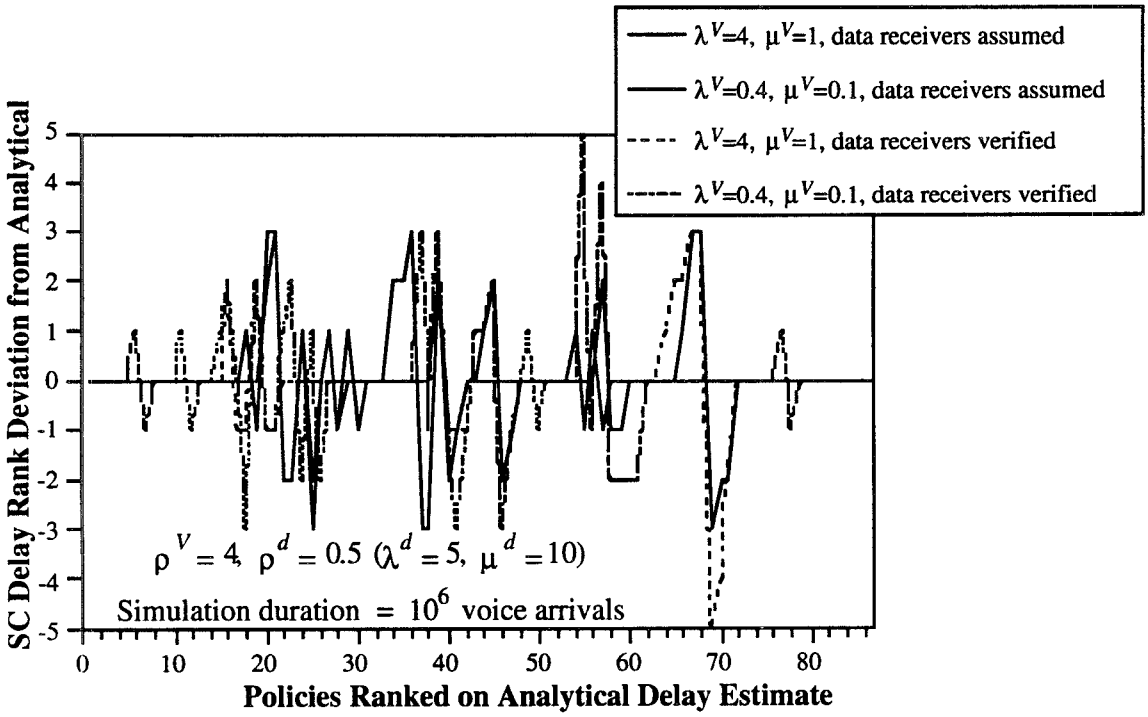
8.2 Data-Packet Delay

In Figs. 17, 18, and 19 we compare the average data delay found by SC simulations (that processed 10^6 voice arrivals) of different control policies to the value computed by our analytical delay estimate. In these figures the x axis represents the 87 best control policies in terms of data delay according to our analytical estimate, with the best policy in position 1. We only plot data for the best 87 policies because the estimated delay is infinite for the remaining policies (delay is infinite, as expected, whenever the offered load is greater than the residual capacity at one or more nodes). As shown in Figs. 17, 18, and 19, various sets of parameters were used, all of which correspond to $\rho^V = 4$ and $\rho^d = 0.5$. For example, the curves in Figs. 17 and 18 were obtained from simulations in which the voice rates were either ($\lambda^V = 4.0$, $\mu^V = 1.0$) or ($\lambda^V = 0.4$, $\mu^V = 0.1$). Since the data service rate μ^d was 10.0 in all the simulations, the expected voice-call duration is 10 times the data-packet length when $\mu^V = 1.0$, and 100 times the data-packet length when $\mu^V = 0.1$. Thus, the total offered load (voice and data) is the same in these simulations, but the parameter sets differ in the voice rates λ^V and μ^V. This is of interest because our analytical delay estimate, which is based on the product-form solution of the voice process, only considers the offered voice *load* (i.e., ρ_j^V). It does not account for the actual values of λ^V and μ^V. By keeping the offered load the same, and varying the voice rate (i.e., vary λ^V and μ^V subject to $\lambda^V/\mu^V = 4$), we can evaluate the impact of this simplification in the analytical delay estimate.

In Fig. 17, we present two views that compare the ordinal rankings obtained from simulation to those obtained from the simplified analytical model for delay discussed in Section 7.1. The rankings from four different SC simulations are plotted. Two of the simulations, those labeled "data receivers assumed," allowed data to be transmitted whenever a transmitter was available at the source node; the availability of a receiver at the destination node was *assumed* guaranteed. Thus, the environment these simulations modeled was similar to that assumed by our analytical delay estimate (which did not require a transmitter-receiver matching). This model is appropriate and adequate for wireline networks where the availability of a transmitter implies that a link, and hence a receiver, is available. Alternatively, we can view these simulations as modeling the case in which it is reasonable to assume that receivers are always available because there are many more receivers than transmitters at each node. In the remaining two simulations, data was transmitted only when a transceiver was *verified* to be available at both the source and the destination nodes. Thus, these simulations modeled an integrated voice/data radio network in which each node has a limited number of transceivers (eight), each of which can function as a transmitter or a receiver, but not both simultaneously.



(a) SC delay rank versus analytical delay estimate rank



(b) Deviation of the SC delay rank from the rank given by the analytical delay estimate

Fig. 17 — A comparison of the ranking of the policies by SC simulation and by the analytical delay estimate

In Fig. 17 (a) the y axis represents the rank assigned to the policy (x axis) as a result of SC simulations. Again, a straight line with unit slope would indicate perfect agreement. Figure 17 (b) gives a different view of the same data; in this figure we plot the deviation of the delay rankings obtained from SC simulations from the ranking given by the analytical delay estimate

(i.e., for policy Ω, Deviation(Ω) = SC rank of Ω – analytical rank of Ω). Since it is difficult to distinguish the individual curves in Fig. 17 (b), we have also included Fig. 18, which shows individual plots of the four curves that are superposed in Fig. 17 (b). The figures show that the agreement for all four simulations, although imperfect, is impressively good. The deviation in ranking among the top 54 policies is never more than three. Furthermore it is interesting that, despite the significant differences between the simulation models (data receivers assumed vs. data receivers verified) and the voice rates ($\lambda^V = 4$ vs. $\lambda^V = 0.4$), which produce significantly different values of delay, the rankings are very similar among the different simulation runs.

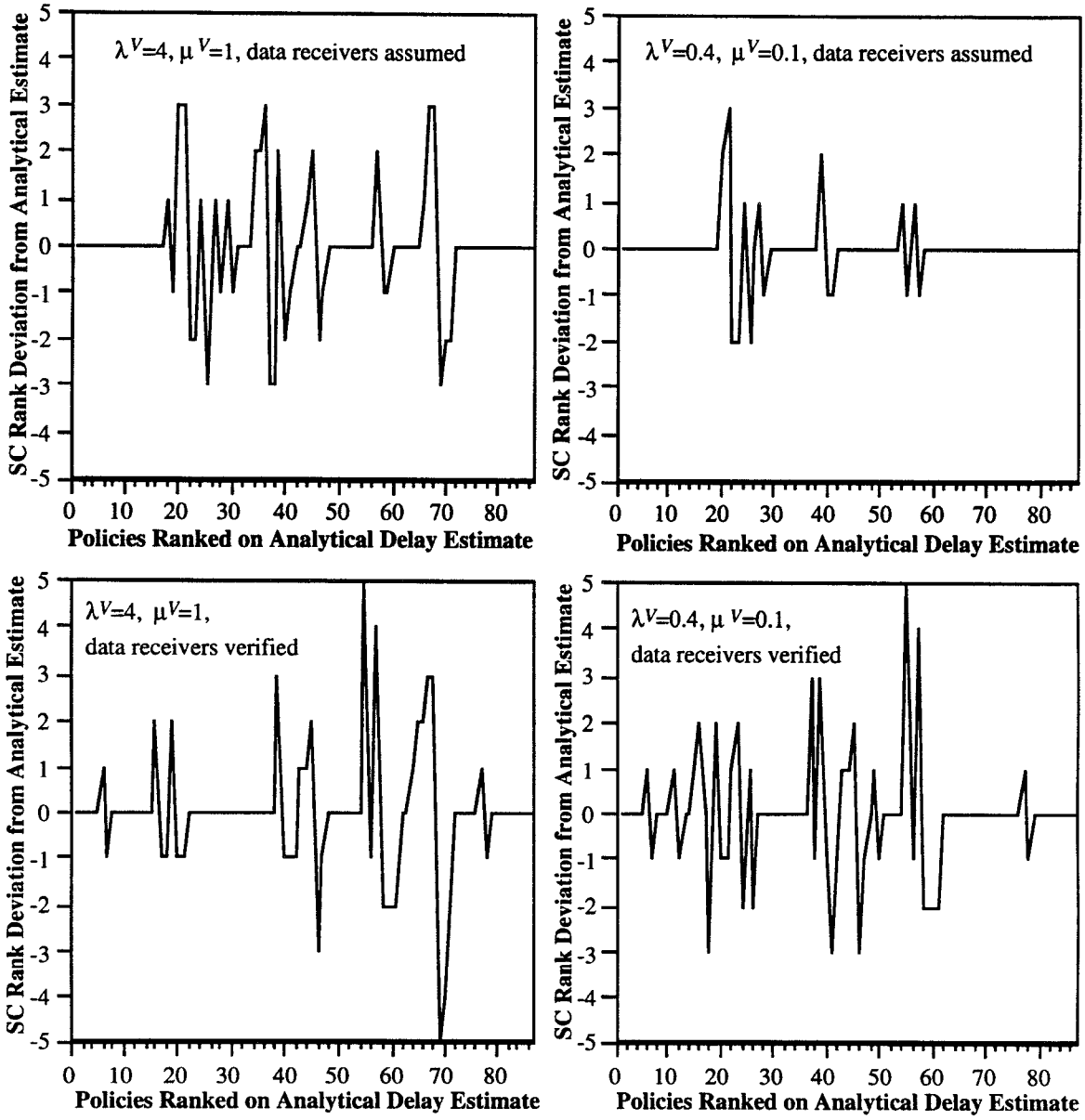
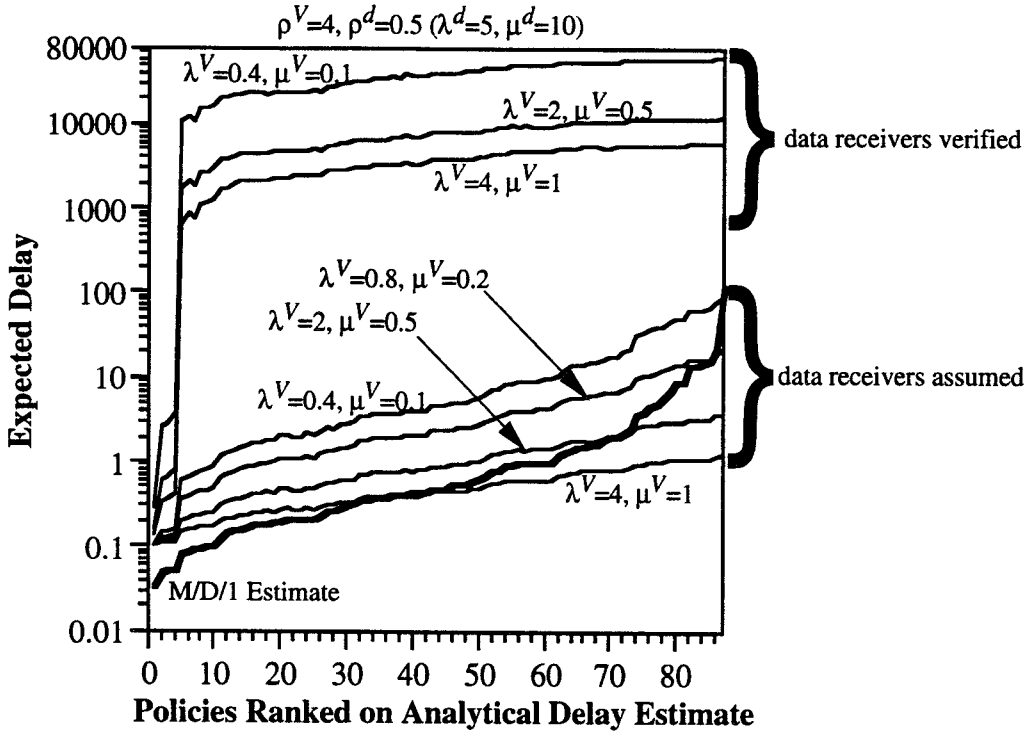


Fig. 18 — Individual plots of SC delay rank deviation from that given by the analytical delay estimate

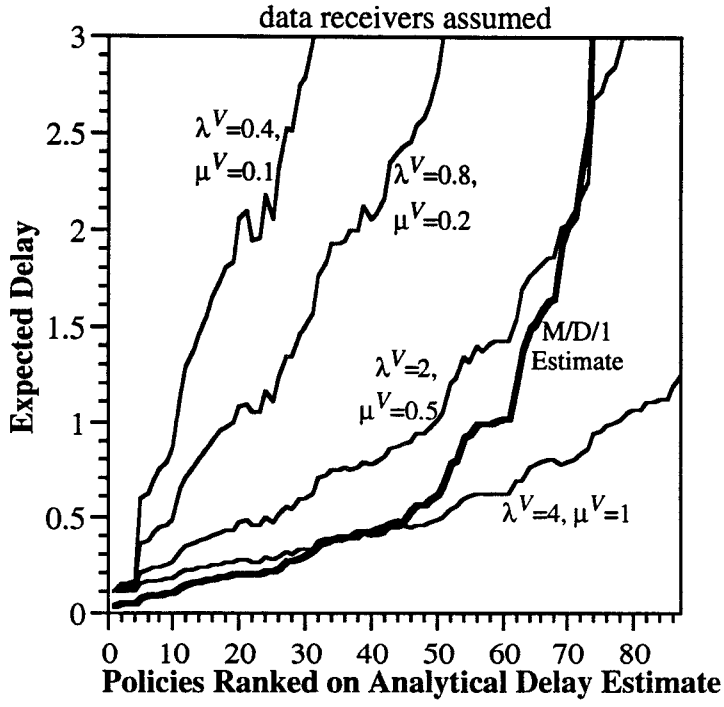
Figure 19 shows the actual delay measured in SC simulations and compares it with the analytical data delay estimate. In Fig. 19 (a), the expected delay is plotted on a logarithmic axis to show complete results from seven simulations, four of which assumed transceivers were available for data reception, and three of which verified the availability of such transceivers. In Fig. 19 (b) the ordinate is expanded and plotted linearly to illustrate the agreement (or lack thereof) between the analytical delay estimate and the values found in the simulations that assumed transceivers were available for data reception. Although Figs. 17 and 18 have shown that the analytical delay estimate does remarkably well in ranking the control policies, Fig. 19 shows that it is not a particularly good estimator of the *actual* delay value. Earlier we discussed the deficiencies in the analytical model. First, it ignores the need for an available transceiver at the receiving node that exists in a typical wireless environment. Also, it ignores the time-varying nature of the number of transceivers that are available. Thus a single analytical solution is shown for the M/D/1 approximation, whereas the simulation results differ for different voice traffic parameters. Although all simulation curves correspond to the same value of ρ, they show that the slower rates of variation in voice state caused by smaller values of λ^V and μ^V (which correspond to longer interarrival times and call durations, and thus in longer periods of heavy voice congestion) result in longer periods in which data packets are forced to remain in queue, and hence, in considerably greater delay.

We have also studied the residual bottleneck data capacity C_b (i.e., the expected communication capacity available for data at the node with the most voice congestion), which does not depend on data traffic statistics, as a performance measure for data traffic. This performance measure is expected to perform better when data traffic is approximately uniform.

Figure 20 compares the ordinal rankings of 120 different policies based on SC simulation data-delay measurements (for $\lambda_i^d = 5$, $\mu_i^d = 10$, therefore $\rho_i^d = 0.5, i = 1, \ldots, 10$, and for $\lambda^V = 4$, $\mu^V = 1$, receivers assumed) to ordinal rankings of the same 120 different policies based on the residual bottleneck data capacity. Note that the simulation rankings agree quite well with the residual bottleneck data capacity rankings throughout the set of all policies, with almost perfect agreement in the first 15 positions.



(a) Logarithmic plot



(b) Linear plot

Fig. 19 — Delay measured in SC simulations compared to the analytical estimate

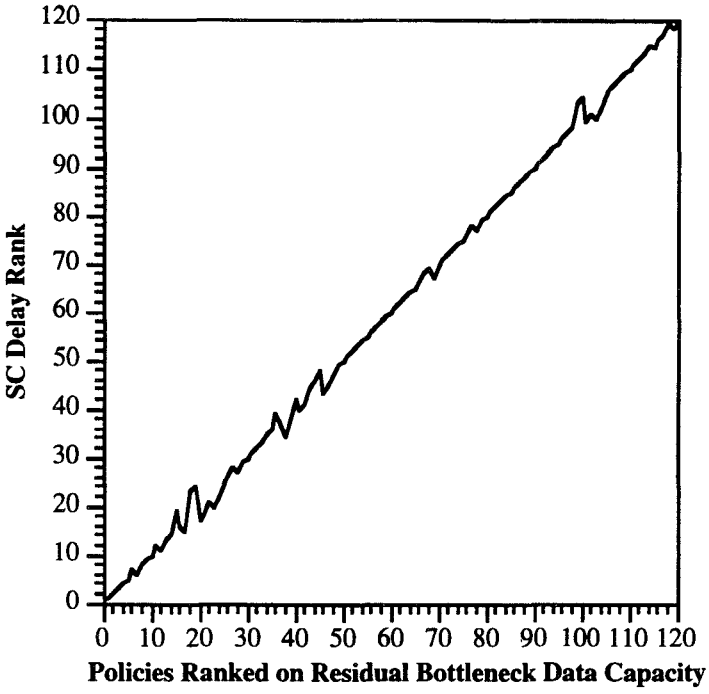


Fig. 20 — Comparison of policy rankings based on residual bottleneck data capacity and the data delay measured in SC simulations

The results of this section suggest the potential value of "ordinal optimization" methods [35]. When we are interested in optimizing performance, the ordinal ranking of the different policies (i.e., the ranking from best to worst) is much more informative than the actual measure of performance under a given policy. In our examples here, Figs. 17, 18, and 19 show that although our analytical delay estimate is inaccurate, it does very well in selecting the best control policy with respect to data delay. For example, although the time constants associated with voice (λ_j^V and μ_j^V) have a major impact on data-packet delay, they only minimally affect the policy ranking. Also, we have observed that the relative rankings of policies are quite insensitive to the requirement (or lack of it) of having a receiver available for each data-packet transmission. We feel that it is especially significant that simple analytical models, such as the M/D/1 queue and even the residual bottleneck data capacity, can provide highly accurate rankings.

The remarkable similarity of the delay rankings obtained using the different models may have interesting, and possibly far-reaching, implications in the study of optimization methods. These results suggest that accurate ordinal rankings can be obtained even when highly inaccurate models (in the sense of predicting actual performance values) are used. The key requirement here appears to be the identification of the key aspects of the model that affect relative performance. In our examples the residual data capacity is of primary importance, while the time constants associated with voice traffic and the availability of receivers at destination nodes is relatively unimportant. Thus, simulation under one set of parameters provides a good indication

of relative performance under other parameter sets (for the same values of utilization ρ_j^V).[12] Moreover, the accuracy of the rankings obtained using the analytical model suggests that simulation may actually be unnecessary to determine the best policy or best set of policies; however, simulation will still be necessary to evaluate the system performance under these policies.

To obtain an intuitive explanation of why crude models can provide accurate rankings, we may view each link as a service system. In general, delay in such systems is a convex increasing function of system load. Figure 21 shows plots of delay versus load for an M/D/1 queueing system and for a "generic" service system with the same capacity. Although the value of delay in the two systems is quite different, the delay curves are both convex increasing functions that approach infinity as the load approaches system capacity. Thus a policy that performs well for the M/D/1 system should perform well for any "similar" system. Actually, the performance measure of interest on a global (network wide) basis is the average delay, where the average is taken over all queues in the system. Since the expected delay at each node is a convex function of load, the overall expected network delay averaged over all nodes is the convex combination of a number of convex functions, resulting in an overall convex and increasing function of the combined total load (i.e., the multidimensional load vector representing the traffic at all nodes). Thus the use of a simplified model at each queue in the network should provide an indication of global performance as well. In the future we expect to verify further the implications of these observations.

We noted earlier that a reasonable performance measure for integrated systems is the weighted sum of voice-call blocking probability and data-packet delay. Unfortunately, our observations on the accuracy of ordinal rankings do not apply to such metrics because the inaccuracy of the delay estimate precludes the formation of a meaningful weighted sum of blocking probability (for which accurate values are available) and delay that would preserve the rankings. However, another reasonable optimization goal, to which our method is applicable, is the minimization of delay, subject to a constraint on voice-call blocking probability. To use this metric, blocking probability would first be evaluated for all policies. Then, among those policies that satisfy the blocking-probability constraint, the minimum-delay policy would be chosen by following the ordinal optimization procedures discussed here.

---

[12] We have observed that this is true provided that the expected voice-call duration is greater than or equal to ten times the data-packet length.

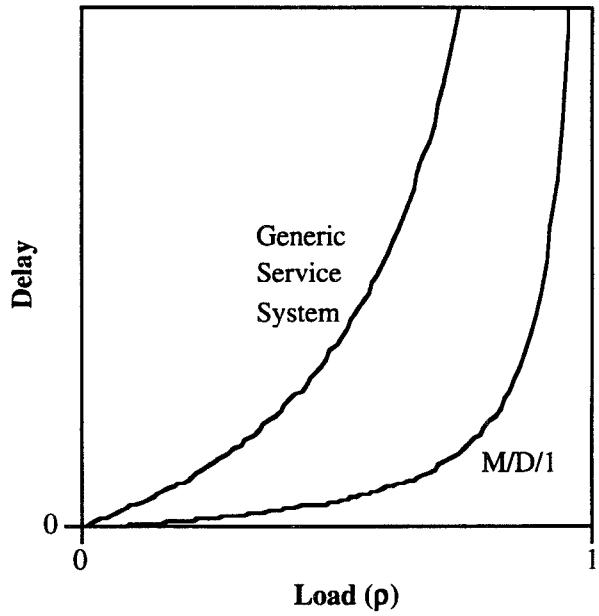


Fig. 21—Delay versus load for an M/D/1 queueing system and a "generic" service system

9 SUMMARY AND CONCLUSIONS

The performance evaluation of complex discrete-event dynamic systems (DEDS), such as communication networks and manufacturing systems, is a difficult task. The ultimate, and perhaps unattainable, goal in studies of such systems is the development of comprehensive models that describe the evolution and performance of DEDS in a manner similar to those that are currently available for continuous-variable dynamic systems (CVDS). Models for CVDS are based on differential and difference equations, and are well understood. However, no major breakthroughs have been achieved in the area of analytical, or even numerical, evaluation of DEDS. Therefore, simulation remains the primary tool for this purpose. However, simulation can be extremely time consuming and consequently expensive, especially in optimization problems in which it is necessary to evaluate system performance for a large number of control policies to find the best one. As a result, the DEDS community has devoted considerable effort to the development of techniques that improve the efficiency of the simulation process. For example, perturbation analysis (PA) methods can provide sensitivity information (and hence a characterization of system performance under a wide range of parameter values) from the observation of a single simulation run. PA and related methods have been applied to a number of communication networking problems such as routing [37-39], link-activation scheduling [19, 40], and buffer occupancy measures [13]. These applications are discussed in more detail in [41].

It is difficult to apply most PA methods to systems with discrete parameters because of the resulting discontinuities in performance measures. Typically, problem-specific techniques must be developed, and they are often highly dependent on intricate mathematical analysis. However, one particular PA-based scheme that is well-suited for systems with discrete parameters is the standard clock (SC) technique [11-13], which is an approach for the parallel simulation of structurally similar DEDS that operate under different parameter values or control policies. Although the applicability of the SC technique is limited to systems with exponential interevent times, it is possible to incorporate certain types of deterministic events into the model, a feature that has permitted us to develop simulation models for integrated voice/data networks. The main feature of SC simulation is that a common clock sequence, consisting of (event time, event type) pairs, is used by a large number of simulations running in parallel, thereby resulting in a significant decrease in the most costly aspect of the simulation process, namely the generation of events.

In this report we have studied the SC approach and have applied it to several examples of DEDS, including queueing systems and integrated radio networks. The first example we studied was the M/M/1/K queue, which we used as a testing ground for the SC approach. Our studies of the M/M/1/K queue were focused on a quantification of the improvement in simulation times that can be achieved as compared with standard brute-force methods, rather than on the performance results of the system being simulated. There was certainly no need to simulate this system to obtain performance results since an exact analytical model is available; however, the availability of an exact analytical solution for this system is precisely the reason we chose it for our initial studies, i.e., it has permitted the validation of our SC simulation model.

Although SC simulation is particularly well suited to simulation on parallel machines, significant improvement can also be achieved on sequential machines. Our study has identified and tracked the time spent in various aspects of the simulation, including event generation and state updating, and has evaluated the simulation speedup that is possible through its use on sequential machines. However, we have shown that the speedup is not linear in the number of sample paths being simulated; in fact, we have shown that an upper limit on the speedup exists, and we have determined an estimate of this limit.

We have also extended the SC simulation model to networks of queues with exponential servers and finite buffers (i.e., queues that would be M/M/1/K in isolation). Simulations of such a network indicate that the SC method scales well with problem complexity (see also, [15, 16]).

We then turned our attention to the study of multihop, circuit-switched voice radio networks. The assumption of Poisson arrival statistics and exponential call lengths, coupled with the desire to determine optimal control policies (which necessitates performance evaluation for a

large number of different policies), makes this problem well suited for SC simulation techniques. It is assumed that there is no queueing of voice calls; the decision to accept or block a call is made on the basis of the system state at the time of the call's arrival. Moreover, the applicability of an exact product-form analytical solution has again permitted the validation of our simulation model. After demonstrating that the `update` function requires only slightly more computation time than in the much-simpler M/M/1/K queue case, despite the much greater complexity of the circuit-switched networking model, we shifted our attention to performance evaluation and the determination of optimal voice-call admission-control policies.

We have demonstrated how our SC model for voice networks can be extended to integrated voice/data operation. In integrated networks we assume that voice calls have priority over data, and thus operate as if data traffic were not present. Unlike voice calls, data packets are queued when they cannot be transmitted immediately. Although it is reasonable to assume that the data-packet arrival process is Poisson, our assumption of fixed-length data packets violates the assumption of exponential interevent times, which is fundamental to the SC method. However, following the approach of Ho et al. [14], we have incorporated deterministic data-packet lengths into our model. A limitation of our simulation model is that the assumption of Poisson arrivals is strictly valid only for single-hop data traffic; this is because the output process of each data queue is no longer Poisson. However, we believe that the use of the Poisson arrival assumption for multihop data traffic will provide a reasonable indication of system performance, and we expect to demonstrate this in the future.

A major factor contributing to the difficulty in finding optimal control policies is the large number of policies that must be examined in many problems of interest. An advantage of the SC approach is that it examines many policies in parallel. However, in our studies "many" has meant 120, whereas a truly exhaustive search of all policies for that problem (i.e., searching over the range of all constraints, not just the three that were varied in our studies) would be prohibitive (see [17]). Although an exact analytical model is available for the circuit-switched voice network, its use in determining optimal policies is practical only for relatively small examples because of the complexity associated with the evaluation of the normalization constant in the distribution of system state. When studying networking problems for which accurate analytical models are not available, as is typically the case, simulation is the primary (and often the only) tool for system evaluation. Thus the use of either simulation or approximate models (which typically must be verified by means of simulation) is needed for most problems of interest, even for the relatively well understood case of product-form networks.

The use of simulation typically requires a long simulation run to obtain accurate estimates of system performance, often making the search for the optimal policy a futile effort. However,

in practice it is often sufficient to determine a *good* policy, rather than the elusive optimal one; the further improvement obtainable through exhaustive search may be insignificant, and may not be achievable within realistic time constraints. Our studies of ordinal optimization techniques have demonstrated that the relative performance of a number of policies can be determined after fairly short simulation runs, providing acceptable performance at reasonable computational expense. The accuracy of the rankings of voice-call blocking probability is especially remarkable in view of the relative insensitivity of performance to policy for the best 80 or so of the 120 policies examined, i.e., it was possible to rank them extremely well even though there was little difference in the actual values of blocking probability. Furthermore, it is interesting to note that a rather crude analytical model for data-packet delay, which is a very poor estimator of the actual delay value, does remarkably well in ranking the control policies.

9.1 Conclusions

In conclusion, we have found the SC approach to be an efficient method for the parallel simulation of DEDS operating under a number of different control policies. In the early stages of this study we applied the SC approach to the M/M/1/K queue, and quantified the improvement in efficiency that can be achieved as compared to traditional brute force methods. We then developed a SC simulation model for multihop, integrated voice/data radio networks. Simulation timing results indicate that this approach scales well with increasing problem complexity. Also, simulated results for voice-call blocking probability agree closely with those predicted by the analytical model.

Often, it is more important to find the optimal control policy than to quantify the performance achieved by implementing that policy. Actually, in practice it is more realistic to search for a good policy that performs almost as well as the optimal one, which may remain elusive because of the vast search space that has to be explored. We have demonstrated that it is possible to achieve a remarkably accurate ranking of the relative performance of the control policies relatively early in the simulation, thus permitting the efficient determination of good, although not necessarily optimal, policies.

Our results suggest that accurate ordinal rankings can be obtained even when highly inaccurate simulation models or analytical approximations are used. We feel that it is especially significant that simple analytical models, such as the M/D/1 queue and the residual bottleneck data capacity, can provide highly accurate rankings. The agreement of the delay rankings obtained using significantly different simulation models and analytical delay estimates indicates that the use of a simple model, which incorporates the system's key aspects, may suffice to accurately rank the relative performance under different parameters and control policies. The

accuracy of the rankings produced by the analytical model suggests that in some applications simulation may not even be needed to determine good control policies. After the set of all control policies is narrowed to a set of good control policies by means of the analytical model, their performance can be evaluated by simulation to determine the best policy. In the future we plan to investigate further the use of such ordinal optimization techniques, possibly in conjunction with other optimization techniques such as adaptive search methods.

Our studies of the SC approach have thus far been limited to simulation examples. We plan to explore the full potential of this method by extending it to the on-line observation and optimization of DEDS examples such as the integrated network model developed in this report.

As in [17], we have not addressed the protocol issues that are associated with call setup, such as the control messages that must be exchanged to request and to verify the availability of network resources needed to support the call. Our focus has been on the performance that can be achieved in an idealized system in which the availability of full state information is available at all nodes, with no time delay or overhead cost. Protocol issues will be addressed in the future, and will take advantage of the insights gained through these studies.

ACKNOWLEDGMENT

The authors gratefully acknowledge Mr. James Ramsey of NRL Code 5521 for his help in developing the parallel machine simulation model described in Section 4.3. The authors have also benefited greatly from discussions with Prof. Christos Cassandras of the University of Massachusetts at Amherst.

REFERENCES

1. Y.-C. Ho and X.-R. Cao, *Perturbation Analysis of Discrete Event Dynamic Systems*, Norwell: Kluwer Academic Publishers, 1991.
2. P. Glasserman, *Gradient Estimation via Perturbation Analysis*, Norwell: Kluwer Academic Publishers, 1991.
3. Y.-C. Ho, "Perturbation Analysis Explained," *IEEE Transactions on Automatic Control*, **33** pp. 761-763, August 1988.
4. R. Suri, "Perturbation Analysis: The State of the Art and Research Issues Explained via the G/G/1 Queue," *Proceedings of the IEEE*, **77** pp. 114-137, January 1989.
5. C. G. Cassandras, *Discrete Event Systems: Modeling and Performance Analysis*, Homewood, IL: R. D. Irwin, Inc. and Aksen Associates, Inc., 1993.
6. Y.-C. Ho and C. Cassandras, "A New Approach to the Analysis of Discrete Event Dynamic Systems," *Automatica*, **19** pp. 149-167, 1983.
7. Y.-C. Ho, X. Cao, and C. Cassandras, "Infinitesimal and Finite Perturbation Analysis for Queueing Networks," *Automatica*, **19** pp. 439-445, 1983.
8. Y.-C. Ho and S. Li, "Extensions of Infinitesimal Perturbation Analysis," *IEEE Transactions on Automatic Control*, **33** pp. 427-438, May 1988.
9. W.-B. Gong and Y.-C. Ho, "Smoothed Perturbation Analysis of Discrete Event Systems," *IEEE Transactions on Automatic Control*, **AC-32** pp. 858-866, 1987.
10. P. Glasserman and W.-B. Gong, "Smoothed Perturbation Analysis for a Class of Discrete Event Systems," *IEEE Transactions on Automatic Control*, **AC-35** pp. 1218-1230, 1990.
11. P. Vakili, "Using a Standard Clock Technique for Efficient Simulation," *Operations Research Letters*, **10** pp. 445-452, 1991.
12. Y.-C. Ho, S. Li, and P. Vakili, "On the Efficient Generation of Discrete Event Sample Paths under Different Parameter Values," *Mathematics and Computation in Simulation*, **30** pp. 347-370, 1988.
13. C. G. Cassandras, J.-I. Lee, and Y.-C. Ho, "Efficient Parametric Analysis of Performance Measures for Communication Networks," *IEEE Journal on Selected Areas in Communications*, **8**-No. 9 pp. 1709-1722, December 1990.
14. Y.-C. Ho, C. G. Cassandras, and M. Makhlouf, "Parallel Simulation of Real Time Systems via the Standard Clock Approach," *Mathematics and Computers in Simulation*, **35** pp. 33-41, 1993.
15. C. M. Barnhart, J. E. Wieselthier, and A. Ephremides, "Use of the Standard Clock to Improve Simulation Efficiency: A Quantitative Study Based on the M/M/1/K Queue," *Proceedings of the 27th Conference on Information Sciences and Systems*, Baltimore, MD, pp. 112-118, March 1993.

16. C. M. Barnhart, J. E. Wieselthier, and A. Ephremides, "Improvement in Simulation Efficiency by Means of the Standard Clock: A Quantitative Study," to appear in *Proceedings of the 32nd IEEE Conference on Decision and Control*, San Antonio TX, December 1993.

17. C. M. Barnhart, J. E. Wieselthier, and A. Ephremides, "Admission Control in Integrated Voice/Data Multihop Radio Networks," NRL/MR/5521--93-7196, Naval Research Laboratory, January 18, 1993.

18. L. Kleinrock, *Queueing Systems Volume 1: Theory*, New York: John Wiley & Sons, 1975.

19. C. G. Cassandras and V. Julka, "Marked/Phantom Slot Algorithms for a Class of Scheduling Problems," *Proceedings of the 31st IEEE Conference on Decision and Control*, Tucson, AZ, pp. 3215-3220, December 1992.

20. V. Julka, C. G. Cassandras, and W.-B. Gong, "Sample Path Techniques for Admission Control in Multiclass Queueing Systems with General Arrival Processes," *1992 Conference on Information Sciences and Systems*, Princeton University, pp. 227-232, March 1992.

21. D. Gross and C. M. Harris, *Fundamentals of Queueing Theory, Second Edition*, New York: John Wiley & Sons, 1985.

22. D. Y. Burman, J. P. Lehoczky, and Y. Lim, "Insensitivity of Blocking Probabilities in a Circuit-Switching Network," *Journal of Applied Probability*, **21** pp. 850-859, 1984.

23. Sun Microsystems, Inc., *SunOS Reference Manual*, May, 1988.

24. P. Bratley, B. L. Fox, and L. E. Schrage, *A Guide to Simulation*, New York: Springer Verlag, 1987.

25. Sun Microsystems, Inc., *C Programmer's Guide*, May, 1988.

26. J. M. Aein, "A Multi-User-Class, Blocked-Calls-Cleared, Demand Access Model," *IEEE Transactions on Communications*, **COM-26**-No. 3 pp. 378 - 385, 1978.

27. C. M. Barnhart, J. E. Wieselthier, and A. Ephremides, "An Approach to Voice Admission Control in Multihop Wireless Networks," *Proceedings of IEEE INFOCOM'93*, San Francisco, CA, pp. 246-255, March 1993.

28. J. E. Wieselthier, C. M. Barnhart, and A. Ephremides, "Optimal Admission Control in Circuit-Switched Multihop Radio Networks," *Proceedings of the 31st IEEE Conference on Decision and Control*, Tucson, AZ, pp. 1011-1013, December 1992.

29. S. Jordan and P. Varaiya, "Control of Multiple Service, Multiple Resource Communication Networks," *Proceedings of IEEE INFOCOM'91*, Bal Harbour, FL, pp. 648-657, April 1991.

30. S. Jordan and P. Varaiya, "Throughput in Multiple Service, Multiple Resource Communication Networks," *IEEE Transactions on Communications*, **39**-No. 8 pp. 1216-1222, 1991.

31. D. J. Baker, A. Ephremides, and J. A. Flynn, "The Design and Simulation of a Mobile Radio Network with Distributed Control," *IEEE Journal on Selected Areas in Communications*, **SAC-2**-No. 1 pp. 226-237, January 1984.

32. A. Ephremides, J. E. Wieselthier, and D. J. Baker, "A Design Concept for Reliable Mobile Radio Networks with Frequency Hopping Signaling," *Proceedings of the IEEE*, **75**-No. 1 pp. 56-73, January 1987.

33. L. Kleinrock, *Communication Nets: Stochastic Message Flow and Delay*, New York: McGraw-Hill, 1964.

34. H. C. Tijms, *Stochastic Modelling and Analysis: A Computational Approach*, Chichester: John Wiley & Sons, 1986.

35. Y.-C. Ho, R. S. Sreenivas, and P. Vakili, "Ordinal Optimization of DEDS," *Journal of Discrete Event Dynamic Systems*, **2** pp. 61-88, 1992.

36. H. A. David, *Order Statistics, Second Edition*, New York: Wiley, 1982.

37. C. G. Cassandras and S. G. Strickland, "Perturbation Analytic Methodologies for Design and Optimization of Communication Networks," *IEEE Journal on Selected Areas in Communications*, **6** pp. 158-171, January 1988.

38. C. G. Cassandras, M. V. Abidi, and D. Towsley, "Distributed Routing with On-Line Marginal Delay Estimation," *IEEE Transactions on Communications*, **38**-3 pp. 348-359, March 1990.

39. W.-B. Gong and H. Schulzrinne, "Application of Smoothed Peturbation Analysis to Probabilistic Routing," *Mathematics and Computers in Simulation*, **34** pp. 467-485, 1992.

40. C. G. Cassandras and V. Julka, "Optimal Scheduling in Systems with Delay-Sensitive Traffic," to appear in *Proceedings of 32nd IEEE Conference on Decision and Control*, San Antonio, TX, December 1993.

41. J. E. Wieselthier, "Discrete Event Dynamic System (DEDS) Models and their Application to Communication Networks," to appear as an NRL Memorandum Report, 1993.