Among the system performance predictive techniques available to the systems engineer are those of theoretical analysis and simulation.

One method of simulation uses the random number generator to simulate the probability distribution of events.

Introduced are principles of random number generator simulation together with examples, the results of which are compared with theoretical results.

Queuing simulation using a random number generator

by R. N. Rechtschaffen

This paper discusses the application of random number generators to the simulation of waiting-line (queuing) phenomena in real-time computer systems. Random number generators produce numbers that are probabilistically distributed in a manner analogous to the throwing of a die. Since one can assume such a randomness in the times of arrivals of users at a waiting line or queue, it is at least intuitively plausible that the arrival times can be simulated by a random number generator. The probabilistic nature of arrival times and service times as well as other time-based phenomena are treated in this paper.

One could say that if everything in the system is regular, the problem would be of no interest. When, for example, each job arrival requires three units of computer time and arrives every four time units, then, in the absence of an initial load, no job must wait. In simulation parlance, the four time units between arrivals of successive jobs are the *interarrival times*. The *transit time*, that is, the elapsed time between arrival and departure, is the three-unit service time.

A typical system handles a stream of arrivals wherein the average interarrival time is four time units, and the average service time may be three time units. This is quite different from the regular case. The concept of *average* suggests that there are interarrival times smaller than four, and service times longer than

three. If the spread in the interarrival times and service times is such that the shortest interarrival time is smaller than the longest service time, then waiting lines form. The extent of the overlap determines the magnitude of the waiting time.

The most direct way of treating such phenomena is by way of mathematical analysis, which uses a body of knowledge known as queuing theory.[1] Analytical and simulative methods are complementary and mutually supportive. Analytical methods typically have the advantages of low setup time, particularly in the simpler queuing cases. Simulation studies by computer require programming, but they can generally handle more complex cases of queuing with greater ease. In designing and analyzing systems, the engineer often uses both performance predictive methods so that one result may confirm the other, thereby giving him added confidence. Setting up a simulation has another advantage over analysis for the systems engineer in that the setup process causes him to think through his design.

There are several distinct types of simulation. As we have mentioned, this paper reviews several cases of simulation that use the random number generator as the driving device. Two other types of simulation are trace-driven and self-driven simulations, wherein actual application programs drive program simulated models of systems. In trace-driven models, a recorded trace of an application program execution drives the simulation model; in the self-driven case, the execution of an application program provides inputs for the simulation model. The effect of simulation is to make system modeling algorithmic as compared with analytic.

We have discussed the probabilistic nature of arrival times, interarrival times, and service times. The history of an arrival's interaction with the system can be summarized by the following quantities: arrival, start, departure, and waiting times. These quantities may be expressed in terms of each other in the following expressions:

Arrival time = prior arrival time + interarrival time
Start time = the greater of either the prior departure time or the current arrival time
Departure time = start time + service time
Waiting time interval = start time − arrival time

Using randomly generated interarrival and service times as inputs, these formulas have been used recursively for two simulations of fifty transactions each to calculate the other times. (Waiting line data have also been computed.) These simulations are summarized in Appendices A and B for use later in this paper.

This paper first discusses the random number generator and gives some examples that illustrate its use. Methods of system analysis and simulation are presented. A problem is worked out in detail to illustrate the manipulation of the random number generator and the validation of results.

Using a random number generator

We first define a computer implementation of a random number generator as a computationally efficient algorithm that generates a sequence of numbers (events) between 0 and 1 so that the numbers are uniformly distributed and stochastically independent. By way of defining such a process, Feller[2] says, "The terms 'stochastic process' and 'random process' are synonyms and cover practically all the theory of probability from coin tossing to harmonic analysis. In practice, the term 'stochastic process' is used mostly when a time parameter is introduced."

The following example particularizes this definition. By subdividing the 0-to-1 interval into six subintervals of equal length the throwing of a die can be simulated. Thus a 1 has occurred if the value generated falls in the first subinterval, a 2 if the value generated falls in the second subinterval, and so forth. For uniform distribution, one might further demand that each outcome, 1, 2, · · ·, 6, should occur with approximately equal frequency in a large number of throws. Uniform distribution of the outputs of a Random Number Generator[3] (RNG) occurs when equal numbers of output values (events) similarly fall in each subinterval.

A sequence of events 1,2,3,4,5,6,1,2, · · · possesses the property of uniform distribution or equal frequencies, but it does not represent a satisfactory simulation of a true die being tossed. The reason is that the successive outcomes are too predictable. That is, successive RNG outputs are not stochastically independent. Reference 3 discusses problems and tests for the stochastic independence of the outputs of a random number generator.

Because the techniques for generating randomness are computational, they actually give pseudo-random outputs. The advantage of generated randomness over pure randomness is that of reproducibility. If a simulation is found to yield a result worthy of special attention, the simulation can be recreated and analyzed in greater depth. It is also possible to exploit the reproducibility in distinct simulations of similar systems by maintaining a high degree of parallelism in the use of the random number generator. This allows the analyst to make matched comparisons.

direct use

Certain simulations that involve the random number generator use its output events directly. For example, in a system having

three types of traffic that follow distinct logical paths, the output of random number generator (RNG) can be used to simulate the decomposition of the traffic into the three categories. Thus, if the fraction of Type 1 traffic is p_1, Type 2 is p_2, and Type 3 is p_3, one way of distributing the traffic is as follows:

Traffic type	Value of RNG
1	$0 \leq \text{RNG} \leq p_1$
2	$p_1 \leq \text{RNG} \leq p_1 + p_2$
3	Otherwise

Since $p_1 + p_2 + p_3$ equals unity, and assuming RNG values are uniformly distributed from 0 to 1, the proper fraction is associated with each type and is preserved in the occurrence of each type within the traffic stream. Similarly, if seventy-five percent of a group of users require a certain mode of service and the remainder do not, this random splitting can also be accomplished by using the random number generator output directly.

transformed use

More typically, numbers generated by the random number generator are transformed to create specified interarrival-time and service-time patterns. A natural occurrence of a random service time within a computer arises, for example, in the timing of direct access storage devices. The *latency time* or rotational delay can vary from zero to a full-rotation time. If the rotation time of a device is 24 ms, then a simple way to transform the RNG values into numbers in the proper range is to multiply RNG by 24.

A simple example of a generic technique that allows RNG values to be transformed to any specified pattern is as follows. If we denote successive RNG values by RNG_i, and the transformation associated with a specified pattern by T, then the values of X_i in the following expression have the specified pattern: $X_i = T(\text{RNG}_i)$

In the rotational delay example, T represents multiplication by a constant equal to 24. As another example of such a transformation, an exponential pattern with a mean (average) of c is given by the following expression where ln is the base of the natural logarithms:

$$T(a) = -c \ln(1 - a)$$

Implementations of such transformations depend on the simulation system used. For GPSS V,[4] the approach to the transformation is based on associating specific (discrete) values or a linear (continuous) interpolation to various subranges of RNG values. As an extension of this example, consider two successive transformations $T'(\text{RNG}) = T(1 - \text{RNG})$. Since both RNG and $1 - \text{RNG}$ are equally good random number generators, the pat-

tern created by T' has the same statistical properties as the pattern created by T. The relationship between successive elements of the two patterns is termed *antithetic*, and the two generated patterns are said to be antithetics of each other.

method of antithetics

The method of antithetics, wherein a pattern is replaced by its antithetic, is one way of illustrating the effect of the random number generator in introducing fluctuations in simulation output. Such a replacement of a pattern by its antithetic in a simulation can have a dramatic effect on the simulation output. This variation in simulation output is not of itself a bad thing. It may mirror the variation in system performance that can be expected. The variation may also indicate that the run size is not sufficiently large to yield a reliable estimate of average system performance.

To place the following example in the proper perspective let us first consider the analytical or queuing theory (equilibrium) result. Consider a single server input as simulated by an exponential arrival pattern so that successive service times are also exponentially distributed. If the system is allowed to stabilize (i.e., reach equilibrium) then the average waiting time T_W can be expressed in terms of the average service time T_S and the server utilization ρ, which is the ratio of T_S to T_A (the latter being the average interarrival time). For the case in which $T_S = 80$, $T = 100$, $\rho = 0.8$, the analytical value of average waiting time is computed as follows:

$$T_W = \frac{\rho}{1 - \rho} T_S = 320$$

Now contrast the analytical result with the results of a random number generator simulation. Expressions for the simulation method of determining the arrival pattern and the service pattern from values found in Appendices A and B are given as follows. These expressions represent antithetical simulations of the analytical case just given.

	Appendix A	Appendix B
Interarrival Times	$-100 \ln(1 - \text{RNG}_{2i-1})$	$-100 \ln \text{RNG}_{2i-1}$
Service Times	$-80 \ln(1 - \text{RNG}_{2i})$	$-80 \ln \text{RNG}_{2i}$
$i = 1, 2, 3, \ldots, 50$		

Using these expressions, the average waiting time for the simulation exhibited in Appendix A is $T_W = 100.38$, and for Appendix B the result is $T_W = 337.44$. These values may be compared with the equilibrium or queuing theory result of $T_W = 320$ previously given.

This gross comparison illustrates the dramatic sensitivity of simulation results to the state of the random number generator.

We show later in this paper the basis for this effect of the antithetic method. In the present case, it is clear that the shortness of the run makes the output statistic (the average waiting time T_w) especially sensitive. This type of analysis, however, is useful in determining the proper run size to reduce the sensitivity to acceptable limits.

A summary observation might be appropriate at this point. The equilibrium result is not to be expected in these simulation cases because short runs depend on initial conditions. In both antithetic runs, the server was initially idle. That is not to say that the equilibrium result does not apply to short periods of time when relatively few users arrive. The important distinction is that the initial conditions in situations where equilibrium can be applied are usually unknown and are the result of prior usage made of the server.

evaluating simulation results

By examining the simulations in detail one gains an insight into why the results differ. Compare the interarrival and service times for users 14 through 23 in Appendix A. These entries represent a set of long interarrival times in conjunction with short prior service times, resulting in no waiting line formation. The opposite effect is observed in Appendix B. Here the same entries represent short interarrival times and long prior service times, resulting in a long waiting line formation. Mathematically, this is to be anticipated if $-\ln(1 - \text{RNG})$ is large, then $-\ln \text{RNG}$ is small. The opposite is also true.

Even though the simulations use different realizations of the arrival and service patterns, the closeness of the realizations to a given standard for both simulations is the same. In both cases, we determine the closeness of the RNG values to a uniform pattern, which is the same as the closeness of $1 - \text{RNG}$ to a uniform pattern (using an arbitrary symmetric measure of closeness).

Several conclusions can be drawn from the examples in the Appendices. The variation in average waiting time that occurs in random number generator simulations is due to the modeling of the chance mechanism, that is, the generator model itself. This dependency of the output on the random number generator is the result of two factors: (1) The value of the numbers generated; and (2) The sequence in which the numbers are generated. The second dependency dictates which of the interarrival times generated is juxtaposed with which prior service time. Thus, the typical simulation procedure is to perform several simulations and average the results as a way of eliminating both dependencies. The process of averaging over several simulations of the same system is in many ways similar to a single long simulation. An equilibrium queuing theory result as a figure of merit corresponding to the performance of the system is thereby justified.

Several short simulations with averaging are often more practical than a long simulation because limited computer time may not allow for an exhaustive performance study of each system, especially when several systems are being compared.

A second simulation of the system by using antithetic patterns can be useful, even though an antithetic simulation is not always as markedly different as in the example given. This is true because both the values and the sequence of the numbers generated are different. Thus, in a statistical context, the output from a given simulation and its antithetic yield negatively correlated estimates of quantities of interest. In the main (but not exclusively) when one output is above the "true" value, its antithetic is below that value. Hence, the average of the complementary antithetic estimates yields an estimate with smaller variation than the average of separate simulation results.

In systems that resemble each other closely, variations due to values and sequences of the generated numbers have a consistent effect. The direction of the fluctuation is usually the same; that is, results of the separate simulations are positively correlated. Exploiting this effect is known as *comparative simulation.*

Comparison of analysis and simulation

We now compare simulation techniques previously outlined with problems for which there are also analytic queuing theory solutions. Treatable by both methods is the general class of single server queuing problems in which the arrival pattern is exponential. It has been observed that if we combine a large number of individual arrival streams, each with its own distinctive pattern, the resultant arrival pattern has exponentially distributed interarrival times. The individual substreams may also have distinctive service requirements, and the resultant service pattern is the weighted average of the individual service patterns. (The arrival pattern is more often observed to be an exponential one than is the service pattern.)

In analytical terms, the formula that yields the average waiting time is part of a larger theorem—named after Pollaczek and Khintchine[1] who first studied it—on the pattern of waiting times as determined by the average interarrival time and the service pattern. This formula can be applied when the server selects the next user to be serviced on any basis other than his service requirements. The average waiting time T_W is computed by the following formula where ρ is the server utilization, the average service time is T_S, and the variance of service time is σ_S^2:

$$T_W = \frac{\rho\, T_S}{2(1-\rho)}\left[1 + \frac{\sigma_S^2}{T_S^2}\right] \qquad (1)$$

The three cases to be simulated and compared with each other and with analytical theory are those in which the service pattern is exponential, constant, and uniformly distributed over the interval 0 to 140. The average interarrival time is 100, and the average service time is 70. The results of applying Equation 1 to the three cases proposed are summarized in Table 1.

For comparison, using random number generator techniques, the average waiting time T_W was computed by five simulations of five-thousand users each. The results are given in Table 2. Average waiting times from queuing theory are displayed at the foot of Table 2, and high (H) and low (L) comparisons of simulation with theory are given with the simulation results. The labeling and, hence, the direction of fluctuation is the same in each of the five simulations for the exponential and uniform service patterns. The only inconsistent labeling is observed in the second simulation with a constant service pattern.

An application of this type of comparison is in the situation where structurally similar systems are being simulated. If one system has a known solution, that system can be used as the basis of comparison, and the labels can be transposed to the other systems. Generally, the transposed labels are correct. The simulation of a system that is structurally similar to an unknown system (and has a known solution) in conjunction with the system that does not have a known solution increases our confidence in the validity of judgments made on the basis of simulation. Further, even when none of the systems has a known solution, the task of picking the best system is simplified on the basis of multiple matched comparisons.

For the simulations in Table 2, consider the ratios of constant and uniform average waiting times to those for the exponential cases. The corresponding analytical theory yields the result that the average waiting time in the exponential case is twice the average waiting time in the constant case. In a given simulation run, however, both the exponential and the constant cases may differ from the theoretical values. A determination of whether simulation output ratios approach two may indicate the validity of both the simulations and the theory. The ratio is a natural means of comparison in queuing theory because all results are scale dependent if they measure time. If one doubles interarrival times and service times, then all the waiting times and the average waiting time double. By computing ratios we eliminate scale dependence. The ratios of successive simulations in Table 2 are given in Table 3.

On a percentage basis, the comparison of the simulations with theory is better for the uniform case than for the constant case. This is due to interaction between arrival and service patterns

Table 1 Theoretical computation of three service patterns

Service pattern	ρ	σ_S^2/T_S^2	T_W	*Ratio with the exponential case*
Exponential	0.7	1	163.333	1:1
Constant	0.7	0	81.666	2:1
Uniform[0,140]	0.7	0.333	108.888	1.5:1

Table 2 Random number generator simulations of three service patterns compared with queuing theory

Simulation	*Exponential*	*Constant*	*Uniform*
1	156.488 L	77.360 L	107.020 L
2	156.588 L	87.726 H	107.902 L
3	136.805 L	76.677 L	97.751 L
4	175.403 H	86.424 H	117.156 H
5	168.443 H	85.828 H	111.384 H
Queuing theory	163.333	81.666	108.888

Table 3 Average waiting time ratios for simulations compared to queuing theory

Simulation	*Exponential/constant*	*Exponential/uniform*
1	2.022	1.462
2	1.785	1.451
3	1.784	1.400
4	2.030	1.497
5	1.963	1.512
Queuing theory	2	1.5

previously noted. In the constant service time case, the results are high or low depending on clustering or nonclustering in the arrival patterns. If the service pattern also allows for variation, a reenforcement or a cancellation may occur within the simulation run. Thus the exponential and uniform patterns react as the theory predicts with respect to each other. This also explains the more consistent high-low labeling in these two cases than in the constant service case.

antithetic simulations

The results of antithetic simulations of each of these three queuing problems are now contrasted with the corresponding straight simulations. One should not consider the antithetic relationship as an asymmetric one; the terms "straight" and "anti-

Table 4 Straight and antithetic simulations

	Simulation	*Straight*	*Antithetic*	*Average*
Exponential service	1	156.488 L	191.990 H	174.239
	2	156.588 L	140.817 L	148.699
	3	136.805 L	170.036 H	153.420
	4	175.403 H	155.268 L	165.335
	5	168.443 H	144.800 L	156.621
Constant service	1	77.360 L	87.485 H	82.422
	2	87.726 H	74.977 L	81.350
	3	76.677 L	80.543 L	78.610
	4	86.424 H	87.146 H	86.785
	5	85.828 H	80.774 L	83.300
Uniform service	1	107.020 L	121.951 H	114.485
	2	107.900 L	95.535 L	101.718
	3	97.751 L	111.825 H	104.788
	4	117.156 H	112.405 H	114.780
	5	111.384 H	102.043 L	106.713

thetic" are used merely for ease of reference. Either simulation may be performed first on the basis of a different initial setting of the random number generator. Comparative simulation results are summarized in Table 4.

According to Table 4, the majority of the simulation run results bracket those of the queuing theory results. The variations from queuing theory for the average of the straight and antithetic runs are smaller than the results themselves. By the usual measure of variation, the sample variance is difficult to see. Consider, however, the maximum and the minimum in the columns to see the reduction in variances resulting from antithetical averaging.

simulation validation

Even in the most thoroughly tested model, the formulator may question its dependency on randomness. It is of central importance that the systems engineer have confidence in the results of a simulation study or else it becomes an empty exercise, and the comparative antithetic approaches assist in establishing this confidence. Further, neither running a simulation for a long time nor repeating the simulation validates the results. Repeating with a change of starting point of the random number generator (random number seed) may be a useful validation technique. This must be implemented carefully to avoid repeating a major part of the old run. Knowing enough about the random number generator and counting the uses made of it can, in many cases, avoid the problem of repetition.

A more powerful suggestion is that of using the antithetic approach. The antithetic in combination with the straight run gives

a better estimator than two separate runs. Of more importance, however, is the fact that the antithetic tends to expose runs that are sensitive to the random number generator output. That is, if a large discrepancy exists between a true value and the result of a simulation, the antithetic is more likely to expose this fact by its being markedly different.

A case study of I/O buffering

the problem

Consider the problem of analyzing a teleprocessing system to determine the requirements for input and output buffer sizes. (The input buffer is designated as COREI and the output as COREO.) The problem system services 50 terminals that connect to the Central Processing Unit (CPU) by 14.8 cps lines. Three types of messages are transmitted at a joint rate of three messages per terminal per minute, and have the following characteristics:

	Type 1	Type 2	Type 3
Probability	0.1	0.8	0.1
Input message length	40	80	20
Input processing time (ms)	5	1	10
Output message length	20	0	100
Additional processing time (ms)	0	0	5-15

In their logical flow through the system, messages wait at terminals until they can be accommodated in COREI. Messages hold the terminal, line, and their assigned storage area in COREI until transmitted. Since several messages may require the channel at the same time, a queue may develop, and thereby increase the amount of time a given message holds COREI. A second channel operation takes place by which the message is written onto an auxiliary storage device. At this point, that message space in COREI is free for other message inputs.

Each of the three message types requires a different processing routine as follows:

- Type 1 retrieves a message from auxiliary storage and schedules its output (1 ms) when storage space is available in the output buffer COREO, and goes to the wrapup routine.
- Type 2 schedules its output (1 ms), waits for space in COREO, and goes to the output routine.
- Type 3 performs the required processing, schedules its output, waits for space in COREO, and goes to the wrapup routine.

The wrapup routine releases the terminal and line back to the user and frees the space in COREO after the message has been transmitted. Message transmission into COREI and out of COREO

requires 68 times the message length in milliseconds on the basis of a 14.8 cps line.

the simulation

Before running the simulation, the systems engineer implements the problem by forming two message groups in a 1:3 ratio. The smaller groups requires 50–150 ms processing time and has an output of 500 characters, and the larger group has 5–15 ms processing time and 100 characters of output as previously given.

One way of simulating such a system is to make the capacity of the buffers large, and, thereby, determine the number of positions actually required. Another approach is to mathematically approximate BLOCKIN-BLOCKOUT buffering, which is the scheme discussed in this paper.

The following analysis highlights the fact that even though most messages require no output, the long outputs generated by the infrequent Type 3 messages dictate that COREO should be larger than COREI. (The technique is known as the sampling of transit-time phenomena.) Our estimate for this problem is that a COREI of 1420 characters and a COREO of 2640 characters is sufficient. The simulation has been run using those buffer sizes as well as using buffer COREI of 3000 and COREO of 3000 characters respectively. Five simulations of 10 minutes of system operation each have been computed for each of these two choices of COREI and COREO. The average simulated transit time is given in milliseconds as follows:

Simulation	COREI = 3000 COREO = 3000	COREI = 1420 COREO = 2640
1	10103	9928
2	9633	10465
3	10679	10442
4	11315	10118
5	10640	10275

This is a rather surprising result in that the smaller buffer storage yields reduced average transit times in four of the five simulations. To examine the results in more detail, the simulated activity at terminal 1 yields the following results in milliseconds:

	COREI = 3000 COREO = 3000			COREI = 1420 COREO = 2640		
Simulation	Number of arrivals	T_W	T_S	Number of arrivals	T_W	T_S
1	26	415	8020	29	1850	8319
2	29	2226	6962	33	2286	6277
3	26	1125	7479	30	1977	7304
4	34	3242	7277	39	3035	6646
5	34	4279	8131	34	2633	6561

These results give a clue to buffer size sensitivity. The changing of the output and input buffer sizes has altered the use of the random number generator, so that the number of arrivals at terminal 1 is different under each buffer-size condition. The types of messages handled in the individual runs are different as well. With the information available at this point, it is difficult to find a systematic pattern relating to buffer sizes and their possible interaction with the random number generator.

The cause of this interaction goes back to the implementation of the problem which results in a random splitting of Type 3 messages into two groups with differing processing times and output message lengths. During the simulation, the arrival times to this section of the model depend on COREI and COREO. Then when the random number generator is used to effect the splitting, its usage affects the subsequent random numbers required by the remainder of the simulation.

Since we recognize that the interaction of the simulation events with the usage of the random number generator may bias the comparison of buffer sizes, we now illustrate the reimplementation of the problem with the planned usage of the random number generator. Basically, Type 3 messages are now broken up into Type 3 and Type 4 messages at the terminals but with the same probabilities as previously given. Thus the new problem conditions are given as follows:

	Type 1	Type 2	Type 3	Type 4
Probability	0.1	0.8	0.075	0.025
Input message length	40	80	20	20
Input processing time (ms)	5	1	10	10
Output message length	20	0	100	500
Additional processing time (ms)	0	0	5-15	50-100

Under the new formulation of the problem, we compute the following simulated average transit times in milliseconds for the buffer storage sizes:

Simulation	COREI = 3000 COREO = 3000	COREI = 1420 COREO = 2640
1	9118	9021
2	10785	10857
3	9044	9093
4	9675	9673
5	10269	10475

The previously noted interaction has thus been removed as illustrated by the following activity of terminal 1 in which times are given in milliseconds:

		COREI = 3000 COREO = 3000			COREI = 1420 COREO = 2640	
Simulation	Number of arrivals	T_W	T_S	Number of arrivals	T_W	T_S
1	33	1783	6477	33	1940	6352
2	25	1228	7697	35	1295	7832
3	26	449	6131	26	478	6087
4	29	756	6338	29	853	6339
5	30	1508	6337	30	1662	6006

From these results, we see the tradeoff that occurs when the buffer size is decreased. With the decrease in COREI, fewer messages are contending in the channel, and the resulting waiting times are reduced. Similarly, a shorter queue develops at the point where COREO is involved. These factors reduce the service time T_S at the expense of increasing the waiting time T_W. There is an overall lengthening of the transit time $T_W + T_S$.

Concluding remarks

Partly by tutorial and partly by example, the use of random number generator techniques for system simulation has been introduced. To gain confidence and to validate results of simulations, comparisons have been made with analytical system performance evaluations.

A test of reasonableness and a variation of the problem as a result of the test also illustrate some principles underlying the random number generator simulation method. Even where the results may be inconclusive, the exercise of problem formulation is useful in understanding the system and in determining the next predictive step to take. More elaborate trace-driven or self-driven system simulation is justified when the engineer knows that analytical and random number methods do not produce valid results. As a final caution, one should be aware of the basically cyclic logic that underlies the validations and comparisons. The quantities that one measures against are produced by the indirect methods just as the quantities being measured. There is no independent standard. Thus one should always apply the test of reasonableness. For example, the first results of the example system simulation might lead the unwary engineer to believe that smaller buffers yield higher throughput. This observation being superficially unreasonable, suggests the reimplementation of the problem. If the initial conclusion were generally true, it should be true under the reimplementation. That it is not, confirms us in our skepticism.

Appendix A

TRANSACTION NUMBER	INTERARRIVAL TIME	SERVICE TIME	ARRIVAL TIME	NUMBER HE SAW IN WAITING LINE ON ARRIVAL	START SERVICE TIME	NUMBER OF ARRIVALS WHILE HE WAS WAITING	DEPARTURE TIME	WAITING TIME
1	14	27	14	0	14	0	41	0
2	61	13	75	0	75	0	88	0
3	24	129	99	0	99	0	228	0
4	113	61	212	0	228	0	289	16
5	272	248	484	0	484	0	732	0
6	73	64	557	0	732	4	796	175
7	3	78	560	1	796	4	874	236
8	75	13	635	2	874	3	887	239
9	0	100	635	3	887	2	987	252
10	54	7	689	4	987	3	994	298
11	88	0	777	4	994	2	994	217
12	187	98	964	2	994	1	1092	30
13	9	21	973	3	1092	1	1113	119
14	53	30	1026	1	1113	0	1143	87
15	241	84	1267	0	1267	0	1351	0
16	30	91	1297	0	1351	0	1442	54
17	133	39	1430	0	1442	0	1481	12
18	100	12	1530	0	1530	0	1542	0
19	471	149	2001	0	2001	0	2150	0
20	28	248	2029	0	2150	0	2398	121
21	128	12	2157	0	2398	3	2410	241
22	105	42	2262	1	2410	2	2452	148
23	99	66	2361	2	2452	1	2518	91
24	31	53	2392	3	2518	0	2571	126
25	145	259	2537	0	2571	1	2830	34
26	27	17	2564	1	2830	2	2847	266
27	44	79	2608	1	2847	1	2926	239
28	66	84	2674	2	2926	1	3010	252
29	239	129	2913	1	3010	0	3139	97
30	235	51	3148	0	3148	0	3199	0
31	72	18	3220	0	3220	0	3238	0
32	431	185	3651	0	3651	0	3836	0
33	30	12	3681	0	3836	0	3848	155
34	295	42	3976	0	3976	0	4018	0
35	69	173	4045	0	4045	0	4218	0
36	32	14	4077	0	4218	1	4232	141
37	75	36	4152	1	4232	0	4268	80
38	282	11	4434	0	4434	0	4445	0
39	143	48	4577	0	4577	0	4625	0
40	175	48	4752	0	4752	0	4800	0
41	1	214	4753	0	4800	0	5014	47
42	202	19	4955	0	5014	0	5033	59
43	133	191	5088	0	5088	0	5279	0
44	752	157	5840	0	5840	0	5997	0
45	26	56	5866	0	5997	1	6053	131
46	43	136	5909	1	6053	2	6189	144
47	89	49	5998	1	6189	3	6238	191
48	53	80	6051	2	6238	2	6318	187
49	31	139	6082	2	6318	1	6457	236
50	77	96	6159	3	6457	0	6553	298

Appendix B

TRANSACTION NUMBER	INTERARRIVAL TIME	SERVICE TIME	ARRIVAL TIME	NUMBER HE SAW IN WAITING LINE ON ARRIVAL	START SERVICE TIME	NUMBER OF ARRIVALS WHILE HE WAS WAITING	DEPARTURE TIME	WAITING TIME
1	202	99	202	0	202	0	301	0
2	77	149	279	0	301	0	450	22
3	151	17	430	0	450	0	467	20
4	38	50	468	0	468	0	518	0
5	6	3	474	0	518	0	521	44
6	65	47	539	0	539	0	586	0
7	336	37	875	0	875	0	912	0
8	63	146	938	0	938	0	1084	0
9	486	26	1424	0	1424	0	1450	0
10	87	191	1511	0	1511	0	1702	0
11	52	464	1563	0	1702	1	2166	139
12	16	27	1579	1	2166	7	2193	587
13	238	114	1817	1	2193	6	2307	376
14	87	91	1904	2	2307	7	2398	403
15	9	34	1913	3	2398	8	2432	485
16	133	30	2046	4	2432	7	2462	386
17	30	75	2076	5	2462	6	2537	386
18	45	153	2121	6	2537	7	2690	416
19	0	13	2121	7	2690	7	2703	569
20	139	3	2260	6	2703	6	2706	443
21	32	152	2292	7	2706	5	2858	414
22	42	71	2334	7	2858	6	2929	524
23	45	45	2379	8	2929	7	2974	550
24	129	57	2508	6	2974	8	3031	466
25	26	3	2534	7	3031	7	3034	497
26	143	128	2677	7	3034	6	3162	357
27	102	37	2779	5	3162	8	3199	383
28	72	34	2851	6	3199	7	3233	348
29	9	17	2860	6	3233	6	3250	373
30	10	59	2870	7	3250	5	3309	380
31	66	127	2936	7	3309	5	3436	373
32	1	8	2937	8	3436	8	3444	499
33	132	156	3069	6	3444	7	3600	375
34	5	71	3074	7	3600	6	3671	526
35	69	9	3143	8	3671	5	3680	528
36	128	145	3271	5	3680	4	3825	409
37	63	80	3334	5	3825	5	3905	491
38	6	160	3340	6	3905	6	4065	565
39	27	62	3367	7	4065	6	4127	698
40	18	63	3385	8	4127	6	4190	742
41	414	5	3799	4	4190	6	4195	391
42	14	122	3813	5	4195	5	4317	382
43	30	7	3843	5	4317	5	4324	474
44	0	11	3843	6	4324	4	4335	481
45	145	54	3988	6	4335	3	4389	347
46	104	16	4092	6	4389	3	4405	297
47	52	61	4144	6	4405	2	4466	261
48	88	36	4232	5	4466	2	4502	234
49	131	15	4363	3	4502	1	4517	139
50	62	28	4425	2	4517	0	4545	92

CITED REFERENCES AND FOOTNOTE

1. W. Chang, "Single-server queuing processes in computing systems," *IBM Systems Journal* **9**, No. 1, 36-71 (1970).
2. W. Feller, *An Introduction to Probability Theory and Its Applications*, 368, John Wiley and Sons, Inc., New York, New York (1964).
3. P.A.W. Lewis, A. S. Goodman, and J. M. Miller, "A pseudorandom number generator for System/360," *IBM Systems Journal* **8**, No. 2, 136-145 (1969).
4. *IBM General Purpose Simulation System V, Introductory User's Manual, Form SH20-0866; and IBM General Purpose Simulation System V, User Manual*, Form SH20-0851, International Business Machines Corporation, Data Processing Division, White Plains, New York 10604.