Simulation Methods for Studying Nonstationary Behavior of Computer Networks

WILLIAM P. LOVEGROVE, JOSEPH L. HAMMOND, MEMBER, IEEE, AND DAVID TIPPER, MEMBER, IEEE

Abstract—**In this paper, we discuss methods for the simulation of the nonstationary behavior of computer networks. Steady-state performance measures, typically computed over time histories, are modified to be applicable to the nonstationary case by redefining the measures as ensemble statistics. The application of well-known ensemble simulation techniques to determining the modified performance metrics from finite ensembles generated by independent replications is given. The simulation methods are illustrated with a comprehensive tutorial example and with a performance study which gives new results for a shared buffer switch.**

I. Introduction

IT is increasingly noted that computer and telecommunication networks must not only have acceptable steady-state performance, but must also perform well under transient or nonstationary conditions [1]–[14]. For example, congestion in computer networks is in general a transient phenomenon, but congestion controls are normally designed and their performance evaluated by a steady-state analysis. Obviously, the performance of congestion control schemes designed in such a fashion may be far from optimal when nonstationary conditions prevail in the network.

Nonstationary conditions occur in computer networks when the statistics of the arrival processes to the network queues or the service processes at the queues vary with time. Nonstationary conditions can be caused by load sharing; changes in routing and flow control parameters; failure of links, nodes, or other network resources; topological changes; network start-up and shut-down; and, most importantly, nonstationary input loads. Under nonstationary conditions, analytical solutions to the queueing system models of computer networks are difficult to obtain.

One approach to determining the nonstationary behavior of networks, used successfully by Van As and others [2]–[8], focuses on numerical methods to solve differential or difference equation models which describe the behavior of the various network queues by time-varying probability distributions or in terms of time-varying mean quantities. These techniques have thus far been primarily limited to Markovian systems and often require the solution of a large number of differential or difference equations.

In light of the difficulty involved in conducting precise analytical studies of the nonstationary behavior of computer networks, simulation is an attractive alternative. Discrete-event simulation has emerged as a basic tool for conducting detailed design and performance evaluation studies of computer communication networks [15]–[19]. Unlike analytical techniques, which typically require a significant number of simplifying assumptions to get a tractable model, simulation allows the network to be modeled to an arbitrary degree of accuracy and detail. Currently, there exist many well-established methods for constructing simulation models of computer networks and performing statistical analysis of the simulation output data [15]–[19]. These techniques have been developed, for the most part, to evaluate the steady-state (i.e., long term) performance of the system under study and many of the techniques must be modified to study the nonstationary behavior.

The approach to nonstationary simulation studies is based upon the well-established concept of generating an ensemble of simulation replications and calculating performance measures as ensemble quantities [9], [10]. General techniques for determining appropriate statistical estimators and calculating confidence intervals for random variables in a nonstationary simulation are discussed in [9], [10]. Several nonstationary simulation studies of the performance of specific computer and telecommunication networks can be found in the literature [11]–[14]. For example, the behavior of several adaptive window flow control schemes subject to nonstationary input loads is studied in [13].

In this paper, appropriate techniques for simulating the nonstationary behavior of networks are identified and discussed in a general and systematic manner applicable to computer networks. The remainder of the paper is divided into six sections as follows. Section II discusses the commonly used performance metrics and, in Section III, these metrics are modified to make them suitable for evaluating nonstationary behavior. Section IV discusses the simulation techniques appropriate for determining the nonstationary performance metrics. The simulation methods are illustrated in Section V with a comprehensive tutorial example and, in Section VI, the methods are used in studying a flow control problem. A final section summarizes the results.

Manuscript received September 29, 1989; revised March 7, 1990 and July 31, 1990. This paper was presented in part at the IEEE Southeastern Symposium on System Theory, Tallahassee, FL, March 26–28, 1989.

The authors are with the Department of Electrical and Computer Engineering, Clemson University, Clemson, SC 29634-0915.

IEEE Log Number 9039273.

II. Steady-State Performance Metrics

The performance of computer networks is typically specified in terms of basic measured quantities of four general types, namely: flow or throughput, number of data units (bits, packets, frames, etc.) stored in devices, point-to-point delay, and utilization. These quantities are normally defined in a manner consistent with measurements that can be made on a physical system as follows:

Flow or Throughput–number of data units passing a specific point in a network per unit time,

Number Stored–number of data units waiting in a device or devices at a specific time,

Delay–time required for a data unit to pass between two points in a network, and

Utilization–fraction of time a device is busy or more generally in a given condition.

Variables defined as above will vary with the time of measurement and will often evidence significant fluctuations more characteristic of local conditions than general network behavior. As is well known, to obtain results with some generality and to make analytical work tractable, one normally works with moments or averages of the first three basic variables. Long time averages are used for physical measurements and these are represented in abstract mathematical models by infinite time averages. For utilization, a long time history is assumed in order to compute the fraction of time in a given condition.

The normally implied time averages are not suitable for studying nonstationary computer network behavior, since by definition nonstationary behavior is time dependent. The next section of this paper attacks this problem by modifying the approach to use ensemble, rather than time, statistics.

Before modifying the approach, however, it is helpful to give mathematical expressions for the normally used measures. Time averages will be denoted as $\langle x(t) \rangle$ and are defined as

$$\langle x(t) \rangle = \lim_{T \to \infty} \frac{1}{T} \int_0^T x(t)\, dt. \tag{1}$$

The commonly used definitions of throughput, number stored, time delay, and utilization are now given.

Throughput: Let $C(t)$ denote the number of data units passing some point after a reference time chosen here to be $t = 0$. Steady-state throughput λ is typically defined as

$$\lambda = \lim_{T \to \infty} \left[\frac{C(T) - C(0)}{T} \right]. \tag{2}$$

If the derivative of $C(t)$ is defined in an extended sense to include impulses (or if the integral of (1) is generalized) then the definition of (2) is equivalent to

$$\lambda = \left\langle \frac{dC(t)}{dt} \right\rangle. \tag{3}$$

Number Stored: The number stored, $S(t)$, is defined as the number of data units waiting in a component at time t. If needed, a vector of stored quantities can be defined to account for the number of data units waiting in any number of components at time t. Whether $S(t)$ is the number waiting in a queue or the number in a queue plus any in service is a detail that can be treated in specific cases. The average number stored is a common steady-state metric and it is given by

$$S = \langle S(t) \rangle. \tag{4}$$

Time Delay: The average time delay, D_{ab}, between points a and b in a computer network, for a fixed path from a to b, is defined as

$$D_{ab} = \lim_{N \to \infty} \frac{1}{N} \sum_{i=1}^{N} D_{ab}^{(i)} \tag{5}$$

where $D_{ab}^{(i)}$ is the delay experienced by the ith data unit over the path and N is the number of data units. The average delay D_{ab} can be used in defining the other types of "average delay," which consider different paths. To preserve symmetry with the other metrics, the average of (5) can be expressed as

$$D_{ab} = \langle D_{ab}(t) \rangle \tag{6}$$

by appropriately defining $D_{ab}(t)$. One suitable definition of $D_{ab}(t)$ is

$$D_{ab}(t) = D_{ab}^{(i)}, \qquad t_i \le t \le t_{i+1} \tag{7}$$

where t_i is the arrival time of the ith data unit. For equality of the averages, the relation

$$\sum_{i=1}^{N} D_{ab}^{(i)}(t_{i+1} - t_i) = \overline{(t_{i+1} - t_i)} \sum_{i=1}^{N} D_{ab}^{(i)} \tag{8}$$

must hold for large N where $\overline{(t_{i+1} - t_i)}$ is the average time between arrivals.

Utilization: The "instantaneous" utilization, $\rho_I(t)$, can be defined as a general binary variable as

$$\rho_I(t) = \begin{cases} 1, & \text{device busy at time } t. \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

As in the case of the other basic variables, the time average of $\rho_I(t)$ is the commonly used quantity. Thus (average) utilization ρ is given by

$$\rho = \langle \rho_I(t) \rangle. \tag{10}$$

Several other useful performance measures can be defined in terms of the basic variables. Some of these include loss rate, cumulative number of data units lost, and power. Also, the probability of a full queue for a finite queueing system, which is not a function of the four basic variables, can be defined in terms of an instantaneous quantity $P_B(t)$ expressed as

$$P_B(t) = \begin{cases} 1, & \text{buffer full at time } t \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

The probability of a full queue P_B is then the time average of $P_B(t)$ or

$$P_B = \langle P_B(t) \rangle. \tag{12}$$

III. Adaptation of Metrics for Nonstationary Behavior

A. Introduction

To study nonstationary behavior, a stochastic process model is used. Recall that a stochastic process $X(w, t)$ maps from an abstract sample space to a time function on $(-\infty, \infty)$, see for example [20]. If w is fixed at w^*, $X(w^*, t)$ is a time function, termed a sample function of the stochastic process. If t is fixed at t^*, $X(w, t^*)$ is a random variable. Of course, if both w and t are fixed, $X(w^*, t^*)$ is a real number. The collection of $X(w^*, t)$ for different values of w^* is termed an ensemble of time functions.

Any physical network operates on a single time history, or sample function. Thus, on one sample function, a performance metric would be represented as $X(w^*, t)$. The time average of $X(w^*, t)$, (i.e., $\langle X(w^*, t) \rangle$) as discussed in the last section, is typically used in steady-state work.

For work with nonstationary networks, the behavior at specific points in time is important and, thus, time averages such as $\langle X(w^*, t) \rangle$ are not appropriate. In adapting the performance metrics used for steady-state operation to nonstationary behavior, the approach is to define the performance metric as a random variable which has significance at a single point in time, or in a small time interval. Thus, in the context developed above, the metric is denoted generically as $X(w^*, t)$. This quantity is a random variable which would take on different values for each sample function. Physically, the different sample functions might be visualized as being represented by identical networks excited by different random arrival patterns from the same statistical distribution. Alternately, one can think of the different sample functions as representing the behavior of a network on successive days.

The maximum useful information concerning $X(w, t^*)$ is its density or distribution function, which would be a function of the time t^*. Here we denote the probability density function of X by $f(X, t)$. In using simulation to study nonstationary systems, it is necessary to generate multiple sample functions of network behavior by using different random number seeds. Each such sample function is termed a "replication" in a simulation study. From this finite ensemble of sample functions, it is possible, in principle, to generate sets of data $\{X(w_i, t^*)\}$ $i = 0, 1, \cdots, N$ for any choices of t^*. From this set of data, a density function for $X(w, t^*)$ could be approximated using a histogram of relative frequencies. It is also possible to use the data to estimate probabilities such as $P\{A \leq X(w, t^*) \leq B\}$ using relative frequencies. Finally, by averaging over the complete collection of data, the expected value $E\{X(w, t^*)\}$ can be estimated. The time-varying expected value will be denoted with a prime as $X'(t)$.

Comparing the types of measures just listed, $X'(t)$ is a simple time function, whereas the estimate of the time-varying probability density function is a function of the random argument w for each value of time. Determination of $X'(t)$ requires much less processing and data storage than the density function. In addition, a time-varying probability density function may have limited use as a performance metric in network design. Probabilities such as $P\{A \leq X(w, t^*) \leq B\}$ could be useful as performance measures since such probabilities are also simple time functions.

In the next section, definitions of the four basic performance metrics will be made so that a variable playing the role of $X(w, t^*)$ is specified and sets of data, represented generically as $\{X(w_i, t^*)\}$, $i = 0, 1, \cdots, N$, can be generated through simulation. This data can then be used to obtain any one of the three types of specific measures discussed above. Most of the illustrative examples in this paper will culminate in the ensemble mean $E\{X(w, t^*)\} = X'(t)$, since for this measure the results can be plotted as a two-dimensional graph and data storage is a minimum.

In formulating the definitions in the next section, an abstract theoretical environment will be assumed. In particular, the time-varying probability density function $f(X, t)$ for the generic performance metric X will be assumed to exist.

B. Performance Metrics for Nonstationary Network Operation

In adapting the steady-state metrics to nonstationary behavior, the average number stored and utilization will be considered first, since they require the least change.

Number Stored: The basic variable $S(t)$, the number of data units stored at time t, is a time-dependent random variable when viewed in the stochastic process context necessary for nonstationary operation. In theory, the probability density function $f(S, t)$ of $S(t)$ exists and it can be used to determine appropriate performance measures. The ensemble average $S'(t)$ is the measure most commonly used.

Utilization: In the steady-state, when ergodicity is assumed, utilization can be defined as either the percentage of time a device is busy or the probability that the device is busy, since both are equivalent.

For nonstationary behavior, the instantaneous utilization $\rho_I(t)$ can be viewed as a time-varying binary random variable with a probability density function $f(\rho_I, t)$. In the case of utilization, the ensemble mean value $\rho_I'(t) = E\{\rho_I(t)\}$ is the quantity of interest since it is numerically equal to the probability that a device is busy.

A similar approach and comments apply to metrics, such as the probability of a full queue which can be related to an indicator function such as $P_B(t)$ as given by (11).

Throughput: In order to obtain a quantity with significance at a single point in time for an abstract model, $\lambda(t)$ can be thought of as the derivative of $C(t)$ with respect to t. Since $C(t)$ is a step function, defining $\lambda(t)$ in this manner requires use of an extended definition of derivative. However, in a practical simulation, $\lambda(t)$ can be

closely approximated by

$$\lambda(t) = \left[\frac{C(t+\delta) - C(t)}{\delta}\right] \tag{13}$$

with δ small but nonzero. In using (13) to approximate $\lambda(t)$, the choice of δ is an interesting issue and this is discussed in Section IV-C.

In an abstract model, $\lambda(t)$ can be regarded as a time-dependent random variable with a probability density function $f(\lambda, t)$. Then, as in the case of the other metrics, a throughput performance metric could be $f(\lambda, t)$ itself, or the ensemble mean $\lambda'(t) = E\{\lambda(t)\}$, or a probability such as $P\{A \leq \lambda(t) \leq B\}$.

Time Delay: Consider the time delay of data unit i moving over a fixed path from a to b in a multinode network. In an abstract model, this delay can be assigned a probability density function which depends on several parameters, namely:

i) the data unit length (which, in turn, depends on the data unit length distribution from which the data unit was selected),

ii) the arrival time t at point a, and

iii) a measure of the states of each network node at the times that the data unit reaches each node on the path $a - b$.

Denote by $D_{ab}(t)$ the delay for an arbitrary data unit arriving at time t to point a. The conditional density function $f(D_{ab}/t)$ is the basis for the time-delay performance metric. As in the other cases, the conditional probability density function can be used directly as the performance metric or used to determine $D'_{ab}(t) = E\{D_{ab}(t)\}$ or $P\{A \leq D_{ab}(t) \leq B\}$.

C. Comments on Nonstationary Behavior

Each of the performance metrics discussed above is time-dependent. For a network which experiences a transient nonstationary period followed by a transition to the steady-state, the nonstationary measures have been defined so that their values approach the values of the steady-state measures as the network moves into the steady-state. This assumes ergodicity in the abstract mathematical model.

For nonstationary behavior, flow into a device is not equal to flow out of the device. Thus, the single variable for throughput $\lambda(t)$ is replaced by two variables termed flow-in $\lambda_{in}(t)$ and flow-out $\lambda_{out}(t)$. Ensemble averages of flow-in and flow-out are given by

$$\lambda'_{in}(t) = E\{\lambda_{in}(t)\} \tag{14}$$

and

$$\lambda'_{out}(t) = E\{\lambda_{out}(t)\}. \tag{15}$$

Little's law does not hold for nonstationary behavior. However, as is shown in [3], the ensemble average of flow-in and flow-out can be related by

$$\lambda'_{in}(t) - \frac{dS'(t)}{dt} = \lambda'_{out}(t). \tag{16}$$

As is the case for steady-state behavior, ensemble average flow-out of a device $\lambda'_{out}(t)$ can be expressed in terms of utilization as

$$\lambda'_{out}(t) = \rho'(t)R \tag{17}$$

where R is the time-invariant average service rate of the device.

IV. Simulation Techniques for Nonstationary Behavior

As noted in Section III, to study the nonstationary behavior of a network, an ensemble of sample functions must be created and this corresponds to generating a large number of independent replications of the system response. As a consequence of the ensemble approach and the need to create nonstationary conditions, a number of differences in simulation techniques arise when compared to the standard steady-state simulation methods. Several of these differences are discussed below.

A. Determination of Performance Metrics

The approach is to determine performance metrics which have significance at a single time point or in a small time interval. It must be possible to measure these metrics on each replication of network behavior, or at least on a subset of such replications. A finite collection or ensemble of measured values is then generated in a simulation study.

In the case of each metric, this finite ensemble of replications can be denoted generically as

$$\{X^{(i)}(t)\}\ i = 1, 2, \cdots, N; \qquad t = t_1, t_2, \cdots, t_k \tag{18}$$

where i indicates a particular sample function and t takes on the values $t_1, t_2, \cdots, t_k$. The finite ensemble is generated by repeated simulation runs with different random number seeds. The expected value of $X(t)$ is estimated by averaging over the finite ensemble to obtain[1]

$$X'(t) = E\{X(t)\} \cong \frac{1}{N}\sum_{i=1}^{N} X^{(i)}(t). \tag{19}$$

The probability $P\{A \leq X(t) \leq B\}$ is estimated, using relative frequency, by determining the number of sample functions N_{AB} for which $A \leq X^i(t) \leq B$. The required probability is then estimated by

$$P\{A \leq X(t) \leq B\} \cong \frac{N_{AB}}{N}. \tag{20}$$

To approximate the probability density, it is necessary to construct a histogram from relative frequency values for a collection of cells. Cells are defined by subdividing the range of values of $X(t)$ with the points $X_1, X_2, \cdots,$

[1] $X'(t)$ will denote both the true ensemble mean and the finite sum approximation.

X_K and determining the numbers of sample functions, N_k, $k = 1, 2, \cdots, K$, for which each of the conditions

$$X_1 \le X^{(i)}(t) \le X_2, X_2 \le X^{(i)}(t) \le X_3, \cdots, X_{K-1} \le X^{(i)}(t) \le X_K \quad (21)$$

holds. The cell relative frequencies are then given by

$$\frac{N_k}{N} \qquad k = 1, 2, \cdots K \quad (22)$$

and these numbers can be used to construct a histogram.

The basic metrics (each playing the role of $X^{(i)}(t)$) for number stored, utilization, throughput, and time delay are now given.

Number Stored: The number stored is determined directly on each sample function so that

$$X^{(i)}(t) = S^{(i)}(t). \quad (23)$$

Utilization: The basic metric for utilization is the instantaneous utilization of (9). Thus,

$$X^{(i)}(t) = \rho_I^{(i)}(t). \quad (24)$$

Throughput: For throughput,

$$X^{(i)}(t) = \lambda^{(i)}(t) = \frac{C^{(i)}(t + \delta) - C^{(i)}(t)}{\delta}. \quad (25)$$

Time Delay: The basic metric for time delay is the delay of data units arriving at point a in an interval about the time t. Thus,

$$X^{(i)}(t) = D_{ab}^{(i)}(t) \quad (26)$$

where $D_{ab}^{(i)}(t)$ is the average delay of data units arriving on sample function i in the interval $[t - \epsilon/2, t + \epsilon/2]$ and ϵ is a small nonzero constant. If no data units qualify for a given sample function, then this sample function does not contribute to the finite ensemble.

B. Confidence Intervals

Confidence interval techniques are often used to quantify the quality of a performance estimate made from simulation data. Here we note that the simulation is structured so that observations across the ensemble of replications at a particular time point are, as nearly as possible, realizations of independent and identically distributed random variables. Thus, the central limit theorem can be used to justify the use of Gaussian statistics to construct confidence intervals on the quantity of interest [10]. A derivation of confidence interval estimators for the basic type of performance metrics discussed above can be found in [10], [15], [16]. As an example, the $100(1 - \alpha)\%$ confidence interval on the sample mean $\hat{X}$ calculated from n observations is given by:

$$\hat{X} \pm t_{n-1,1-\alpha/2}\hat{V} \quad (27)$$

where $\hat{V}$ is the unbiased sample variance of the n observations and $t_{n-1,1-\alpha/2}$ is the upper $(1 - \alpha/2)$ critical point obtained from the student-t distribution with $n - 1$ degrees of freedom. One can readily see that the size of the confidence interval is dependent upon the number of observations n (i.e., number of replications) and the variance of the observations $\hat{V}$. Thus, the size of the resulting confidence intervals for a given number of replications can vary widely depending upon the particular system involved and the parameter being measured. For example, consider an $M/M/1/5$ queue with a service rate of $\mu = 1$ and a mean arrival rate of $\lambda = 0.5$. We assume that the queue has attained steady-state when the system is observed at time $t = 0$. This is accomplished by choosing the initial conditions for the simulation (i.e., the number in system) to be near the steady-state point. A short overload burst in the arrival rate of $\lambda_{max} = 1.5$ occurs from $t = 25$ to $t = 50$ s. For an ensemble of 200 independent replications, the resulting estimates of the time-varying delay and average number in the system are shown in Fig. 1. Note the difference in the size of the confidence intervals relative to the estimate itself in the two cases, even though they are based on the same number of observations. This is because the variance of the queue length distribution is smaller than that of the delay distribution.

Note that, if one assumes that enough observations have been gathered to form an accurate estimate of the sample variance, then the only way to reduce the size of the confidence interval on a mean performance measure will be to increase the number of replications. Determining ahead of time the number of replications required to obtain a desired accuracy is a difficult problem particularly when a complex system of queues is involved. A good general discussion of the problems involved in achieving a desired accuracy can be found in [10].

In a general context, the accuracy is problem-dependent. In some cases, only the general shape of the transient is desired in order to study the general response of the system to a particular input. In this case, large confidence intervals are acceptable and only a few replications may be needed. At other times, such as when comparing two system designs, the user may want a high precision and small confidence intervals are required. For these reasons, it is desirable if the simulation can be used in an interactive mode in which results are displayed graphically as they are produced and the user can interrupt the simulation if desired.

Software for a simulation package with an interactive mode, which graphically produces performance curves and confidence intervals on-line, has been written and used to produce the results in this paper. The user of the package can observe the confidence intervals in estimating the system response as more and more replications are added and stop the simulation when a desired accuracy is obtained. Alternately one can determine the number of replications through use of relative precision rp [15]. Relative precision for a confidence interval is defined as the ratio of the confidence interval half-width to the mean of the quantity being estimated, and is given by

$$rp = \frac{t_{n-1,1-\alpha/2}\hat{V}}{\hat{X}}. \quad (28)$$

Note that one can specify a desired relative precision for each performance measure at every time point of interest.

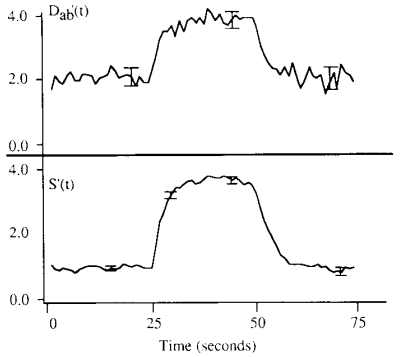


Fig. 1. Comparison of confidence intervals for estimated ensemble delay and average number in system for an $M/M/1/5$ queue experiencing a traffic burst from $t = 25$ to $t = 50$ s.

The relative precision can be estimated after each replication and the simulation can be stopped after the desired relative precision on all confidence intervals is attained. The ensemble simulation approach also allows one to determine the number of observations that actually lie within the confidence interval, thereby determining the actual coverage of the confidence interval.

C. Issues in Measurement of Performance Metrics

The basic performance measures used in determining ensemble statistics are defined in Section IV-A. Note from the definitions in (25) and (26) that throughput and delay are dependent on the choices of δ and ϵ. The basic metric for time delay is the average of the delay of data units arriving in an interval of length ϵ. If ϵ is too large, the $D_{ab}^{(i)}(t)$ arriving at slightly different times in the intervals could be significantly different and the time resolution of delay measurements would be poor. A similar comment could be made considering $\lambda^{(i)}(t)$ and δ since, in this case, $\lambda^{(i)}(t)$ is effectively the flow rate averaged over the interval δ.

The choice of δ and ϵ depends on the time rate of change of $\lambda^{(i)}(t)$ and $D_{ab}^{(i)}(t)$ which are not known in advance of the simulation. An order of magnitude bound on the rate of change of any system parameter, however, can be determined by considering the event rate for the whole network under simplifying assumptions. If all of the events (arrival, service, etc.) taking place in the system are assumed to have exponentially distributed interevent times with average rate r_i, then the overall average system event rate is $r = \Sigma_i r_i$. No system parameter in the simulation can change more rapidly than r on the average, including $\lambda^{(i)}(t)$ and $D_{ab}^{(i)}(t)$. Thus, a bound on the average time between arrivals is $1/r$ and the transient may be missed if ϵ or δ is significantly more than this value.

From the point of view of sampling accuracy, if the average rate of a variable assumed to be Poisson is λ, the expected number of arrivals in N replications during a time interval δ is $N\lambda\delta$. Thus, in order to have at least one arrival in all N replications $\delta > 1/N\lambda$, with a similar result for ϵ.

Another issue arising in the estimation of performance metrics involves the time delay which, as noted earlier, is determined by measuring the delay of data units. One characteristic of this definition that should be noted is that time delays cannot be observed until the data units have reached their destinations. As a consequence, to obtain the estimated delays for data units arriving at the end of a specified interval, it is necessary to continue the simulation until all the packets have reached their destination. If the system delays are long, this will significantly increase the computer run time required for the simulation. Another problem arises when the number of arrivals is small, so that some replications may not contain arrivals. In such a case, the number of replications required to obtain accurate results is greatly increased.

For example, consider an $M/M/1/5$ queue subject to a service interruption. During the service interruption, the queue fills rapidly and the ensemble blocking probability quickly reaches unity, causing arrivals to be lost. Thus, delay measurements during this interval become inaccurate because of a lack of arrivals accepted into the queue from which to measure delay. As an illustration of this problem, an $M/M/1/5$ queue was simulated with a service interruption ot 25 s and the results are shown in Fig. 2. The lack of arrivals during the latter part of the service interruption produces a dramatic increase in the size of the confidence interval, even though the number of replications is large.

D. Issues in Choosing Time Sample Points

For the type of discrete event simulation used to study computer networks, every significant event must be generated. The sequence of time points at which variables are sampled, however, can be chosen independently of the generation of events. Of course, system variables only change when events occur.

In selecting the time points at which to sample the simulated states, a tradeoff between data collection cost and accuracy of the estimates is sometimes required. Sample points too widely spaced may not adequately represent the transient effects sought, while sample points too closely spaced may require excessive data collection.

A collection of N replications of the system typically produces N samples from which to base an estimation of state variables at each sample time. The spacing of time samples is independent of the number of replications. Hence, there is minimal computation cost in decreasing the sampling interval other than increasing the volume of data produced.

It should be noted that the sample points need not be equally spaced. They may be more closely spaced in regions where it is known that the system parameters will vary rapidly.

E. Arrival Processes

Well-known techniques exist for generating a variety of common interarrival distributions at constant arrival rates [15], [19]. A few methods exist for creating nonstationary

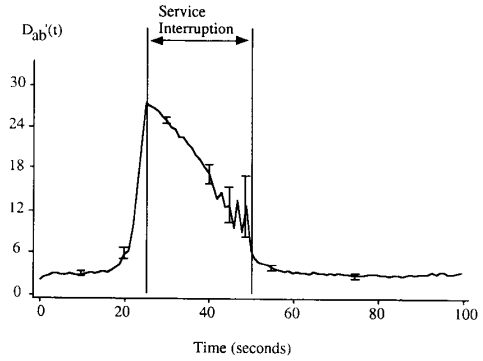


Fig. 2. Demonstration of large confidence intervals due to insufficient arrivals upon which to base an estimate of ensemble average delay. Based on 5000 replications with $\lambda = 0.5$, $\mu = 1$, and $\epsilon = 0.5$.

arrival processes. These include Poisson, Gaussian, and Bernoulli processes. One common pitfall is to merely choose the interarrival time of the data units at the currently desired time-varying arrival rate. This, however, will produce too few arrivals in the time immediately after an increase in the arrival rate. A general approach that correctly generates nonstationary arrivals for some processes, such as a Poisson arrival process, is the thinning method [15]. The idea is based on Bernoulli splitting of a process and the approach is to create arrivals at the maximum arrival rate encountered during the simulation and then to accept the arrivals probabilistically into the system with a time-varying probability $p_{in}(t)$, to achieve the proper arrival rate at time t. If we denote the desired mean arrival rate at time t by $\lambda(t)$, and the maximum arrival rate during the performance study by λ_{max}, then the acceptance probability is defined as

$$p_{in}(t) = \lambda(t)/\lambda_{max}. \tag{29}$$

Unfortunately, the thinning algorithm is computationally expensive since, during periods of low arrival rates, much computation is "wasted" on generating interarrival times for packets which are discarded.

An alternative method proposed by Tran-Gia [24] requires a potentially difficult inversion of the integral of $\lambda(t)$ but avoids "wasting" random numbers.

F. Random Number Generation

Generally, simulations make use of so-called pseudo-random number sequences and a wide variety of generator algorithms for such sequences are available [15], [25]. When collecting ensemble statistics as opposed to time averages, multiple simulations of the system are performed and it is required that each replication be independent. It is common to require hundreds or thousands of replications to achieve a useful accuracy in the results. Thus, long sequences of random numbers are required and the quality of the random number generator must be carefully considered. In order to avoid correlation among the replications, a common technique is to use the last number generated in a replication as a seed to begin the next replication [15]. Also, it is important to avoid repetition in the random number sequence, hence, the period of the random number generator should exceed the sum total length of the sequences used in all replications.

V. Comprehensive Tutorial Example

As an illustration of the utility of the metrics and simulation techniques discussed in the previous sections, consider their application to a system consisting of a single $M/M/1/K$ queue. This queue is chosen because analytic results for transient ensemble average queue length are available. The arrivals occur according to a Poisson process with mean $\lambda'(t) = \lambda$ and the service times have an exponential distribution with mean μ. There is storage space available for up to K jobs in the system, including the one in service if any. The system is empty and idle at $t = 0$ when the input is first applied. The parameters are chosen to have the following values: $K = 5$, $\lambda = 0.5$, and $\mu = 1.0$.

A. Number in the System

Fig. 3 plots one replication of the number in the system $S(t)$ for the $M/M/1/5$ queue. Also shown in the figure is the time average of $S(t)$ over 100 s and the theoretical steady-state mean number in the system. Fig. 4 gives ensemble averages of $S(t)$ over 50, 500, and 5000 replications. The figures show the 95% confidence intervals if these averages are used to approximate the ensemble mean of $S(t)$, which is denoted by $S'(t)$. Fig. 4 also includes a theoretical curve for $S'(t)$ as computed using the methods of Morse [21] from the stated starting conditions. It should be noted that this sequence of plots in Fig. 4 can be obtained on-line using the sequential procedure of Section IV to obtain the confidence intervals.

As discussed in Section IV, the averages over the multiple replications are carried out in the simulation by sampling each replication at a sequence of times $t_1, \cdots t_k$ and computing the averages at each time from the data thus obtained to construct $S'(t_i)$, $t_i = t_1, \cdots t_k$. The curves in the figures result from connecting the points with straight lines.

Fig. 5 plots a state occupancy histogram for this same system at various points in time, again based on 5000 replications.

B. Delay

To illustrate determination of delay through simulation, the $M/M/1/5$ queue is modified by assuming that the server is inactive for the interval $t = 25$ to $t = 35$ s. The average input rate λ to the queue, as well as the other parameters, remains at the values used in Section V-A.

Fig. 6 gives the ensemble average of the number in the system, as computed from 5000 replications, as a function of time. As would be expected, the number in the system increases when service is interrupted and decreases gradually toward a steady-state value when service is resumed.

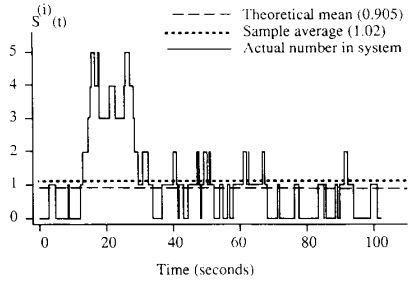


Fig. 3. One replication of an $M/M/1/5$ queue plotting the number in the system as a function of time. Also plotted are the theoretical mean number in the system, as averaged over time, and the average number in the system in this replication over the first 100 s.

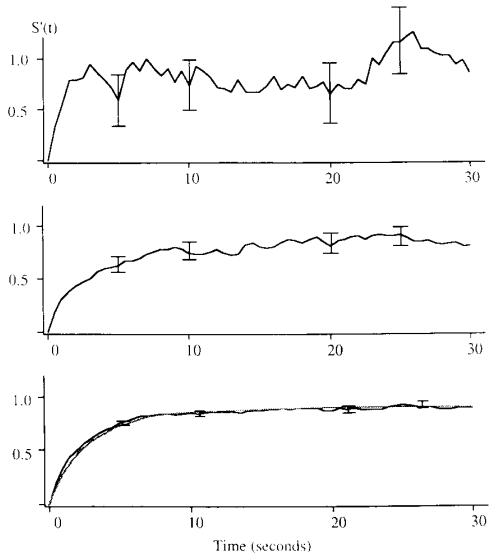


Fig. 4. Ensemble average number in the system with 95% confidence intervals based on 50, 500, and 5000 independent replications of an $M/M/1/5$ queue starting empty and idle with an arrival rate of $\lambda = 0.5$. The dashed line represents the theoretical ensemble average.

The estimated ensemble average delay $D'_{ab}(t)$ for packets entering the simulated system is also given in Fig. 6. As discussed in Section IV, the curve is constructed by averaging the delays of all packets arriving in a small interval $[t - \epsilon/2, t + \epsilon/2]$ to estimate $D'_{ab}(t)$.

The shape of the delay curve can be justified intuitively as follows. A packet which arrives just prior to the service interruption has a high probability of not completing service before the interruption. If service is not completed, the packet must wait the entire duration of the service interruption before its service can be continued. Packets arriving at the instant service is interrupted experience the largest delay. In the example, the service interruption (10 s) is long compared to the service time for one packet (1 s). Thus, delay decreases for arrivals occurring later and later in the period of service interruption. When service is resumed, a short time is required to reduce the backlog of waiting packets.

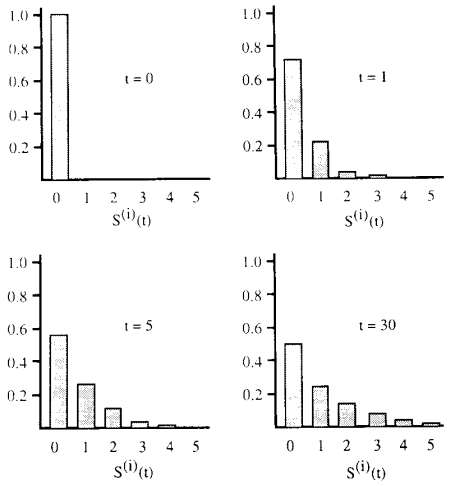


Fig. 5. State occupancy probability histograms showing relative state occupancy at various times for an $M/M/1/5$ queue starting empty and idle and experiencing $\lambda = 0.5$. Based on 5000 replications.

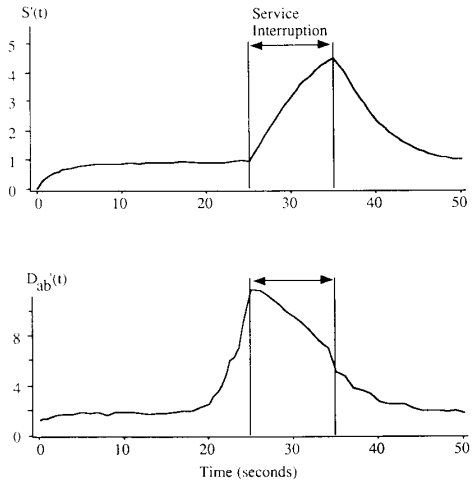


Fig. 6. Ensemble average number in the system and ensemble average delay in an $M/M/1/5$ queue experiencing a service interruption. Based on 5000 replications with $\lambda = 0.5$, $\mu = 1$, and $\epsilon = 0.5$ s.

C. Throughput

A traffic burst is applied to an $M/M/1$ queue with infinite (very large) storage to illustrate the measurement of average throughput $\lambda'(t)$ using simulation. Estimates of ensemble averages computed from 1000 replications are shown for number in the system and flows in and out in Fig. 7. The derivative of the number in the system is also plotted in Fig. 7. From the figure, one can clearly see the validity of (16).

VI. Transients in a Classic Flow Control Problem

Transient behavior in a classic flow control problem, apparently first proposed by Kleinrock and Gerla [22] and later discussed by Bertsekas and Gallager [23] among oth-

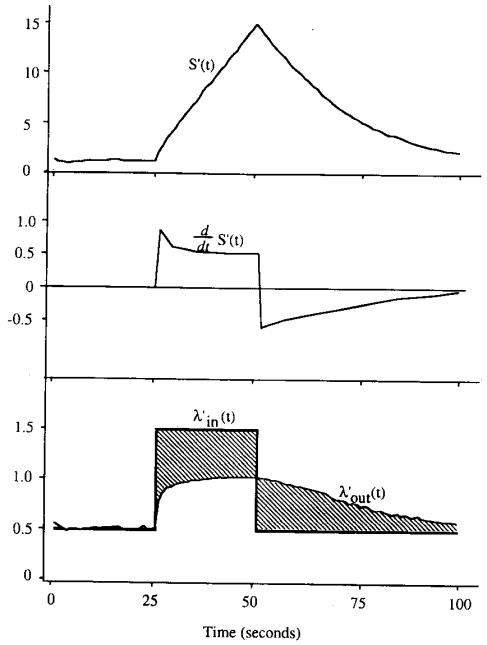


Fig. 7. Ensemble average number in the system in an $M/M/1/\infty$ queue and its derivative, along with the ensemble average flows into and out of the queue. The shaded region represents the difference between the ensemble average flows in and out. The correspondence between this difference of the flows and dS'/dt should be noted [3].

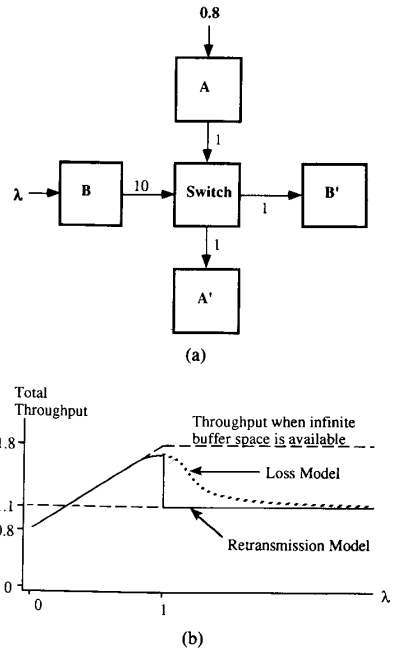


Fig. 8. (a) The shared switch used in a classic flow control problem to study congestion; (b) the steady-state throughput of the switch (from [22]).

ers, is discussed in this section as representative of a class of problems. The configuration is shown in Fig. 8(a).

A switch is shared by two pairs of hosts engaged in communication. All transmission lines have a rate of 1 packet/s, except for the line from source B which has a rate of 10 packets/s. The average traffic rate from A to A' is a constant 0.8 packet/s. The average traffic rate from B to B' is a variable parameter and is denoted by λ. Both arrival streams have a Poisson distribution. The switch contains two output queues, one for each destination. However, the buffer space available in the switch is to be shared by these two channels and is finite. Packets arriving when all the buffers in the switch are full may be discarded (the loss model) or retransmitted at a later time by the source (the retransmission model). For simplicity, packets were placed in the queue at the source for retransmission as soon as they were lost and acknowledgment delays were not modeled.

Steady-state analyses of this problem can be found in the references cited above. The result for total throughput to both destinations A' and B', as a function of the average arrival rate λ into B, is given in Fig. 8(b) (a slightly revised version of the figure found in [22]). For either model, as λ approaches unity, the output queue from the switch to destination B' grows until it fills all of the available buffer space. When this happens, a dramatic drop in throughput results, as shown in the figure.

The result in Fig. 8(b), determined from steady-state behavior, does not give the full picture in many practical applications. The problem, in many cases, is the response of the system to short-term transient overload bursts. During an overload burst, the system may be unable to handle all of the traffic and delays will temporarily increase. Once the overload is over, the system has the capacity to deliver all of the built-up backlog of traffic and eventually the system can return to steady-state. The designer of flow control systems is faced with determining the length of time the overload condition exists, the maximum delays that are experienced, and the buffer sizes required to prevent packets from being lost. These issues require a transient analysis of the system.

To carry out a transient analysis, the retransmission model is considered. The available buffer space is chosen as 20. An overload burst is modeled by applying an input from source B with $\lambda = 0.5$ and a burst of $\lambda = 2$ between $t = 50$ and $t = 80$. Thus, the ensemble average arrival rate experiences an increase as a rectangular pulse with a duration of 30 s to a value four times its steady-state. This arrival pattern, although specific, is felt to be a typical worst-case scenario and has been used in several other studies [2], [7], [13]. The simulations were begun in the fixed state of the system being initially empty.

The transient behavior of the system is captured in three sets of curves obtained using 5000 independent replications. The large number of replications results in very

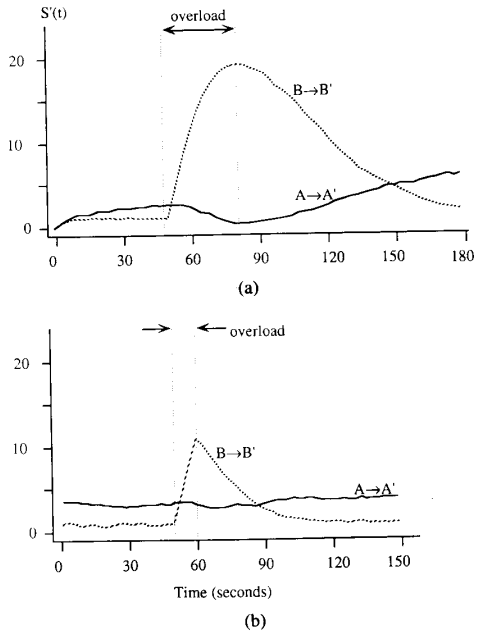


Fig. 9. Ensemble average number in the system for each queue as the switch experiences an overload traffic burst duration of (a) 30 s and (b) 10 s.

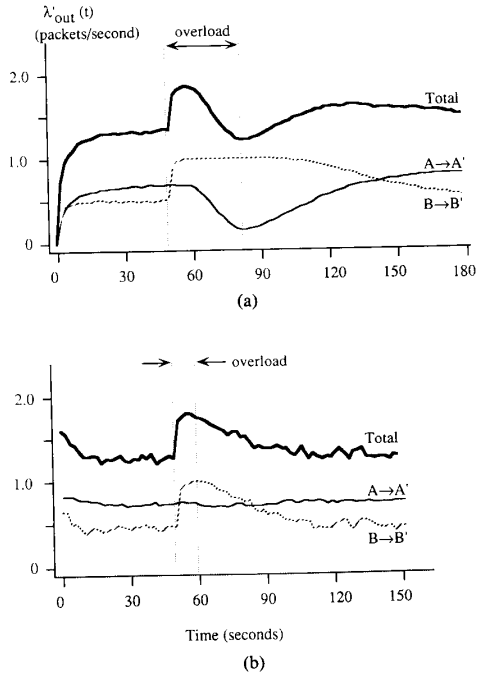


Fig. 10. Ensemble average throughput out of each queue (to A' and B') as the system experiences an overload traffic burst duration of (a) 30 s and (b) 10 s, $\delta = 0.5$ s.

small confidence intervals and, for the sake of clarity in the graphs, they are not shown here. Fig. 9(a) shows the number of packets waiting and in service for delivery to destinations A' and B' as a function of time. Fig. 10(a) shows the flows to the two destinations and the total flow to both destinations, also as a function of time. Finally, Fig. 11(a) shows delays to the two destinations for packets arriving at the times indicated on the abscissa of the curve.

Fig. 9(a) shows that the number of packets stored in the BB' path builds up to a maximum and then decreases, as could be expected. For the path AA', the number of packets stored initially decreases and then increases as the steady-state is again approached. The behavior of the flows, as given in Fig. 10(a), shows fluctuation during the transient with a return to almost steady-state conditions roughly 150 s after the transient overload has ceased. The delay curves in Fig. 11(a) show the surprising condition that abnormal delays persist after the flows have returned to approximately steady-state values. In fact, the transient lasts approximately 500 s.

A heuristic explanation of the persistence of long delays is the following. The source contains a queue of packets which are awaiting retransmission, and although the queue in the switch is small the retransmission queue is quite large. New arrivals must await the transmission of all of the packets in the retransmission queue before getting service from the switch and, hence, they experience a long delay.

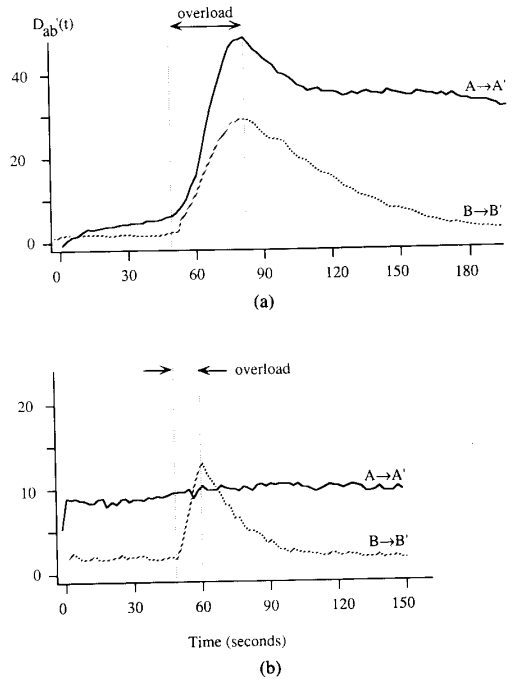


Fig. 11. Ensemble average delay experienced by traffic from each source as the system experiences an overload traffic burst duration of (a) 30 s and (b) 10 s, $\epsilon = 0.5$ s.

A better understanding of this system's behavior is gained by realizing that the steady-state throughput of channel A is 0.8, so the excess capacity available to work off a backlog of packets is only 20%, or 0.2 packets/s. When the A' server is idle, as it is during congestion, packets are being accumulated in the queues at a rate equal to the arrival of 0.8 packets/s. Once the server resumes service, this accumulation of packets can only be reduced at 0.2 packets/s. Recovery from the transient is thus approximately four times as long as the overload itself. Loss of throughput on this channel A is extremely costly in terms of the time required to recover from the resulting congestion.

If the traffic overload is short enough that the traffic from source B does not fill the switch and cause congestion, there would then be negligible loss of throughput on the A channel and little effects on its queue sizes or delays. The overload traffic from B would be served quickly and the system could be expected to return to steady-state much more rapidly than in the case above.

To check this out, a simulation was run for the overload reduced to a duration of 10 s. The results are shown in Figs. 9(b)–11(b). Compare these results to those shown in Figs. 9(a)–11(a) for the 30 s overload. As expected, the system recovers much more quickly from this overload. Although the return to steady-state on channel B is not dramatically quicker, the effect on channel A is greatly reduced. As a result, the system returns to normal much more quickly. In the previous case, the system takes about $500s$ after the end of the transient. For the reduced overload, the system has returned to normal in about 50 s.

A precise measure of the duration of the transient is given by the recovery time, which is defined as the time it takes the system response to return to within a small percentage of the steady-state value (typically $\pm 2\%$). The recovery times of the system for various overload levels and durations, as obtained from simulations similar to the ones above, are plotted in Fig. 12 versus overload pulse duration, with maximum overload as a parameter. In each case, a sharp "knee" occurs in the curve at the duration at which the system becomes congested. Overloads longer than this critical limit produce dramatically longer times for the system variables to return to their pre-overload values.

The results of a number of simulation runs were used to obtain the plot in Fig. 13, which gives the time to congestion (the knee of the curves in Fig. 12) versus the maximum overload λ_{max}. The asymptote at $\lambda_{max} = 1$ is a result of the fact that there is no congestion if $\lambda_{max} < 1$ and, therefore, the time to congestion is unbounded.

Points on the curve in Fig. 13 can be approximated with a steady-state analysis, which gives a partial verification of the transient simulation results. A steady-state approximation of the system behavior can be determined as follows. Under moderate loading, each output queue behaves approximately like an $M/M/1/20$ queue. The steady-state mean number in the system of an $M/M/1/20$ queue is given by a known result, which for $\mu = 1$ can be

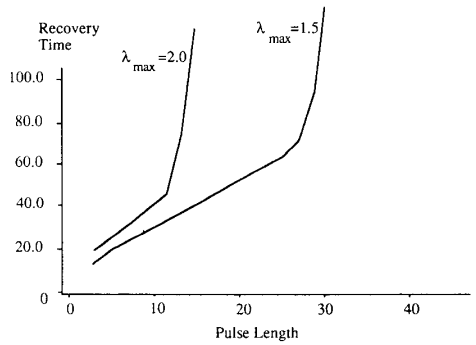


Fig. 12. Recovery time from an overload as a function of the overload pulse duration and of the overload arrival rate λ_{max}.

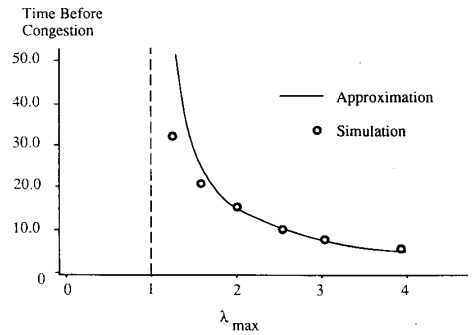


Fig. 13. Time interval before congestion occurs as a function of the overload arrival rate λ_{max}.

written as

$$S = \lambda \frac{1 - 21\lambda^{20} + 20\lambda^{21}}{(1 - \lambda)(1 - \lambda^{21})}. \tag{30}$$

For channel A with constant arrival rate $\lambda = 0.8$ packets/s, this gives $S_A = 3.80$ packets. For the steady-state load from source A of $\lambda = 0.5$ packets/s, this gives $S_B = 1$ packet.

For a given overload arrival rate $\lambda_{max} > 1$, we have excess arrivals and therefore a queue buildup at a rate of approximately $(\lambda_{max} - 1)$ packets/s. Congestion will begin when queue B reaches system size $(20 - S_A) = 16.2$ packets. Queue B is already in steady-state with mean $S_B = 1$, as calculated above. The number of new arrivals which will lead to congestion is, therefore, approximately 15.2 packets. The time to build up to the 15.2 packets, which is approximately the critical overload length beyond which the system will become congested and take an exceptionally long time to recover, is given by

$$t = \frac{15.2}{\lambda_{max} - 1} \text{ s}. \tag{31}$$

The time t computed above is plotted versus λ_{max} in Fig. 13 for comparison to the transient simulation results.

To further extend this example, consider a simple flow control mechanism added to the system. Lam and Reiser

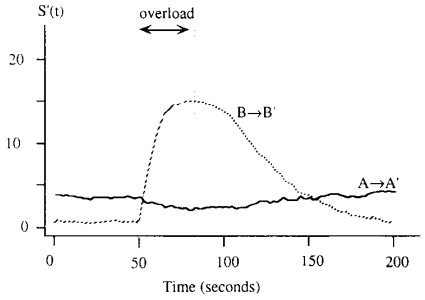


Fig. 14. Ensemble average number in the system for each queue as the switch experiences an overload traffic burst of duration 30 s with a buffer limit of $S_I = 15$.

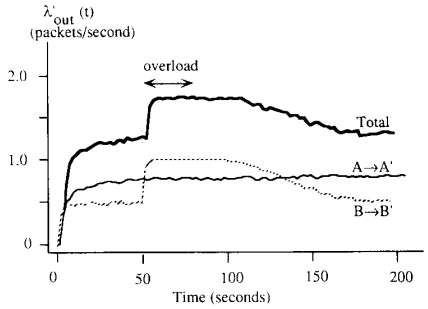


Fig. 15. Ensemble average throughput to each destination as the system experiences an overload traffic burst of duration 30 s with a buffer limit of $S_I = 15$, $\delta = 0.5$ s.

[26] propose an input buffer limiting scheme, which works well in this situation. Suppose that the total amount of buffer space which can be utilized by either queue is restricted to an amount smaller than the switch total buffer capacity. In other words, although the switch contains a total capacity of 20 packets, neither queue is allowed to utilize more than some limit S_I of them, where $S_I < 20$. This guarantees that the other queue will always have at least $20 - S_I$ buffers available. The problems with blocked packets while a server is idle are thus eliminated. In this example, the queue for destination A' is guaranteed $20 - S_I$ buffers.

The number S_I can be as small as $S_T/2$, which in this case equals 10. With this limit, the buffer space in the switch is equally partitioned between the two queues and there is no buffer sharing and, therefore, no advantage. They behave as two independent $M/M/1/10$ queues. If the limit is lowered below 10, some buffers will always be unused. There is no reason to use such a buffer limit, which could only hurt system performance.

Choosing a limit of $S_I = 15$, for example, the 30 s overload is applied with this buffer limit. The plots of system size, throughput, and delay are shown in Figs. 14–16 for this case. Comparing to Figs. 9–11, the increase in throughput during the overload is obvious, as well as the dramatic decrease in recovery time. The improvement in throughput was predicted by the steady-state analysis, but the recovery time is dependent upon the nature of the overload and can only be quantified by a transient study, as was done here.

The recovery time for a variety of choices of S_I and a variety of overloads is shown in Fig. 17, the results of simulation. From these results, the optimum S_I buffer limit can be chosen so as to minimize the effect of the overload on the system. Note that, in this case, a transient analysis provides better information upon which to base design decisions. Interestingly, in this example, there is little benefit in buffer sharing at all. The system performs almost as well under these particular circumstances, with the buffers evenly divided between the two queues. If the specific input pattern chosen for this example was known to be the prevalent traffic pattern in this system, the switch could be simplified by implementing independent buffers.

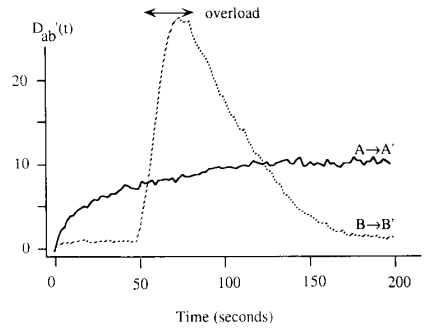


Fig. 16. Ensemble average delay experienced by traffic from each source as the system experiences an overload traffic burst of duration 30 s with a buffer limit of $S_I = 15$, $\epsilon = 0.5$ s.

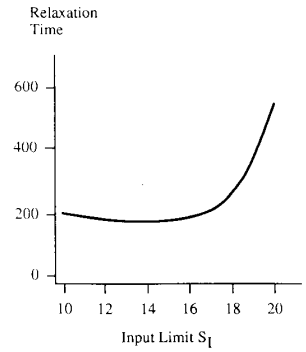


Fig. 17. Recovery time from an overload of duration 30 s as a function of the input buffer limit S_I.

Although the transient study presented above has, of necessity, been for a specific input pattern, the results are typical of a wider class of systems and the methods can be used to get specific results for other parameter choices.

VII. Conclusions

This paper develops a systematic approach to the simulation studies of nonstationary computer network behavior. The basic approach is to redefine the normally used time average performance measures as ensemble statistics. Although this approach is not new, the paper de-

velops each of the basic performance measures in a unified manner and specifies how each variable can be estimated in a simulation study.

The measurement of each of the basic metrics is illustrated in a simulation study for a tutorial example. The metrics defined and the simulation methods given are shown to be computationally feasible. A simulation study of a nontrivial shared-buffer switch is presented in detail and new results are obtained. The methods presented in this paper have been implemented in a software package, which makes it possible to construct a sequence of replications allowing observation of the confidence intervals on-line and termination of the simulation when the confidence intervals are satisfactory.

Several issues needing further study have been discussed, namely rules for determining the time sample points to gather statistics, methods for precisely determining the time interval size for accurate observation of the throughput and delay (i.e., δ and ϵ), and variance reduction methods to reduce the number of replications required for a specified confidence interval accuracy. Nevertheless, the methodology discussed in this paper provides a viable approach to simulating nonstationary network behavior and provides the network designer with a valuable tool.

REFERENCES

[1] J. Hammond and J. Spragins, "Rapidly reconfiguring communication networks: Definition and major issues," in *Proc. IEEE INFOCOM 87*, San Francisco, CA, 1987, pp. 202-206.

[2] H. Van As, "Transient analysis of Markovian queueing systems and its application to congestion control modeling," *IEEE J. Select. Areas Commun.*, vol. SAC-4, pp. 891-904, Sept. 1986.

[3] D. Tipper and M. K. Sundareshan, "Numerical methods for modeling computer networks using nonstationary conditions," *IEEE J. Select. Areas Commun.*, this issue, pp. 1682-1695.

[4] S. Tripathi and A. Duda, "Time-dependent analysis of queueing systems," *INFOR*, vol. 24, no. 3, pp. 199-219, 1986.

[5] A. Weiss and D. Mitra, "A transient analysis of a data network with a processor-sharing switch," *AT&T Tech. J.*, pp. 4-6, Sept./Oct. 1988.

[6] D. Mitra and A. Weiss, "The transient behavior in Erlang's model for large trunk groups and various traffic conditions," in *Proc. 12th Int. Teletraffic Cong.*, Torino, Italy, June 1-8, 1988, pp. 1367-1374.

[7] H. Van As, "Dynamic behavior of network access congestion control mechanisms," in *Proc. IEEE INFOCOM'86*, Miami, FL, Apr. 1986, pp. 20-29.

[8] J. Zhang and E. J. Coyle, "Transient analysis of quasi-birth-death processes," *Stochastic Models*, vol. 5, no. 3, pp. 459-496, 1989.

[9] P. Glynn and D. Iglehart, "Simulation methods for queues: An overview," *Queueing Syst.*, vol. 3, pp. 221-256, 1988.

[10] P. Welch, "The statistical analysis of simulation results," in *Computer Performance Modeling Handbook*, S. Lavenburg, Ed. New York: Academic, 1983.

[11] A. Garcia and W. Shaw, "Transient analysis of a store-and-forward computer communications network," in *Proc. 1986 Winter Sim. Conf.*, 1986, pp. 752-759.

[12] P. Hanselka, "Subcall-type overload simulation: A method to adapt the call terminating rate to rate of accepted calls in the case of transient processes," in *Proc. 12th Int. Teletraffic Cong.*, Torino, Italy, June 1-8, 1988, pp. 1265-1269.

[13] S. Pingali, D. Tipper, and J. Hammond, "The performance of adaptive window flow controls in a dynamic load environment," in *Proc. IEEE INFOCOM'90*, San Francisco, CA, June 5-7, 1990, pp. 55-62.

[14] B. Wallström and H. Vogt, "Transient behavior of simple overload strategies for SPC switching systems—An analytic approach," in *Proc. 12th Int. Teletraffic Cong.*, Torino, Italy, June 1-8, 1988, pp. 371-378.

[15] A. Law and W. Kelton, *Simulation Modeling and Analysis*. New York: McGraw-Hill, 1982.

[16] A. M. Law, "Statistical analysis of simulation output data," *Oper. Res.*, vol. 31, no. 6, pp. 983-1029, Nov./Dec. 1983.

[17] J. Kurose and H. Mouftah, "Computer-aided modeling, analysis and design of communication networks," *IEEE J. Select. Areas Commun.*, vol. 6, pp. 130-145, Jan. 1988.

[18] K. Pawlikowski, "Steady state simulation of queueing process: A survey of problems and solutions," *ACM Comput. Surveys*, vol. 22, no. 2, pp. 123-170, 1990.

[19] J. Banks and J. Carson II, *Discrete-Event Simulation*. Englewood Cliffs, NJ: Prentice-Hall, 1984.

[20] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*. New York: McGraw-Hill, 1965.

[21] P. Morse, *Queues, Inventories, and Maintenance*. New York: Wiley, 1958.

[22] M. Gerla and L. Kleinrock, "Flow control: A comparative survey," *IEEE Trans. Commun.*, vol. COM-29, no. 4, Apr. 1980.

[23] D. Bertsekas and R. Gallagher, *Data Networks*. Englewood Cliffs, NJ: Prentice-Hall, 1987.

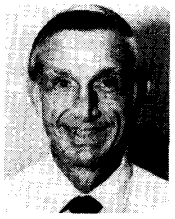
[24] P. Tran-Gia, "Simulation of instationary processes for performance evaluations of switching systems," in *1st Euro. Simulation Cong.*, Aachen, Germany, pp. 362-367, Fachbericht 71, Springer-Verlag, 1983.

[25] D. E. Knuth, *The Art of Computer Programming, Vol. II*. Reading, MA: Addison-Wesley, 1969.

[26] S. S. Lam and M. Reiser, "Congestion control of store-and-forward networks by input buffer limits," *IEEE Trans. Commun.*, vol. COM-27, pp. 127-134, 1979.

William P. Lovegrove received the B.S. degree in physics from Bob Jones University, Greenville, SC, in 1984, the M.S. degree from Clemson University, Clemson, SC, in 1986, and is currently pursuing the Ph.D. degree in computer engineering from Clemson University. His doctoral research is focused on studying transient behavior in communication networks using simulation.

Since 1988, he has served as an Instructor in Electrical Engineering at Bob Jones University.



Joseph L. Hammond (S'52-A'53-M'58) received the S.B. and S.M. degrees in electrical engineering from the Massachusetts Institute of Technology, in 1952 and the Ph.D. degree from the Georgia Institute of Technology, in 1961.

He is currently Professor of Electrical and Computer Engineering at Clemson University, where he has been since 1985. Previous employment includes Lockheed-Georgia Co., 1984-1985, as a full-time Consultant, and Georgia Institute of Technology, 1955-1984, where his last title was Professor of Electrical Engineering. He is engaged in teaching and research in the areas of computer networks and computational techniques. He is the author (with P. J. P. O'Reilly) of *Performance Analysis of Local Computer Networks* (Reading, MA: Addison-Wesley, 1986) and a number of papers and technical reports. He holds a patent on an analog-to-incremental-digital converter.

Dr. Hammond is a member of Sigma Xi and Tau Beta Phi.

David Tipper (S'78-M'88), for photograph and biography, see this issue, p. 1695.