MODELING AND SIMULATION

FOR TELECOMMUNICATIONS NETWORKS

Victor Frost, IEEE Senior Member

Telecommunications and Information Sciences Laboratory
Electrical Engineering and Computer Science
University of Kansas
Lawrence, Kansas

and

Benjamin Melamed, IEEE Fellow

NEC USA, Inc., C&C Research Laboratories
Princeton, New Jersey

1. INTRODUCTION

Users of communications networks would like to have the best of all worlds. They would like the network to be always available, to exhibit ideal performance (i.e., be transparent to end users), to provide ubiquitous connectivity, and to improve continually, so as to accommodate additional users and new services, applications, and technologies. The fundamental goal of network designers and managers is to meet such customer expectations.

In the post-divestiture environment, design and management decisions for telecommunications networks are thus being driven by ever-increasing customer expectations in a competitive marketplace. Properly made decisions call for predictions of network performance; decisions based on poor predictions may adversely affect network users, resulting in severe business consequences. Performance prediction transforms a given network architecture and a description of user demands for network resources into a set of performance predictions that directly relate to the user's perceived quality of service. Such metrics may include response time, throughput, and the probability that user requests for network resources are turned down. Because the demands for network resources are random and the operation of the typical network is highly complex, deriving most network performance predictions is an intricate procedure. Analytical techniques, computer simulation, projections from existing experience, and experimentation are all used to evaluate and compare network designs and protocols. Regardless of the prediction methodology used, design and management decisions must often be made with incomplete knowledge of impending user demands, as well as considerable uncertainty as to the evolution of existing technologies and, by implication, the system itself.

Mathematical analysis has been used to good effect since the earliest days of telephony, notably by the Danish telephone engineer A.K. Erlang, who pioneered the study of queueing phenomena around the turn of the century. Analytical techniques use a mathematical model, typically based on queueing theory [1-14], to represent the network. Often the average resource holding time, e.g., packet lengths and link capacities, and the frequency of user demands (packet arrival rate) are used to characterize the network load. The desired performance metrics are calculated from the load and model specifications. Speed is a major advantage of analytical models, as results can usually be obtained within minutes. However, these models are often based on restrictive assumptions regarding the nature of the load and the operation of the network. For example, many models assume that the times between requests for network resources are independent random variables with exponential probability density functions (pdf), and that the requests are served in a first-in-first-out (FIFO) order. Even though modeling assumptions often limit the validity and accuracy of these models, analytical techniques are quite useful in the early stages, since they permit a quick derivation of rough qualitative system behavior. The theoretical approach can often provide an initial approximate prediction of system performance and shed light on the basic qualitative tradeoffs that must enter into design and management decisions. (See the references above for a detailed discussion of analytical techniques used in telecommunications networks.)

Performance can also be predicted by extrapolating from past experience. When small variations on a base network are called for, an extrapolation technique may be quite useful. However, the relationship between the network load and the performance metrics of interest can be highly nonlinear. It is common for such systems to work in almost a binary mode with no graceful degradation, i.e., the system operates satisfactorily below some threshold of network demand, but small changes in load above that threshold cause a very large change, resulting in unacceptable system performance. For example, using the simplest queueing model (an M/M/1 queue [6]) to represent a statistical multiplexer with an output link rate of 56 Kbps and an average packet length of 1024 bytes, one finds that a change in load from 20% to 25% only causes about a 7% increase in delay, whereas a change in load from 70% to 75% results in a 20% increase, while a corresponding change from 85% to 90% causes a 50% increase in delay. Thus, special care must be taken when attempting to project future network performance based on past history.

In principle, performance prediction can also be determined from direct experimentation on real-life operating networks. In most cases, this is not a practical approach. Networks are too large and expensive to build solely to determine their performance. Furthermore, direct experimentation on an operating network can damage or crash the network and, at the very least, will likely disrupt the network, inconveniencing paying customers.

Monte Carlo computer simulation is a flexible performance prediction tool in widespread use in science and engineering. Its flexibility stems from the fact that it consists in the main of a computer program that "behaves" like the system under study. Unlike analytical models that often require many assumptions too restrictive for most real-world networks, simulation modeling places relatively few restrictions on the classes of systems under study. For communications networks, developing a simulation program requires: 1) modeling random user demands for network resources; 2) characterizing network resources needed for processing those demands; and 3) estimating system performance based on output data generated by the simulation. The operation of communication networks can be conveniently described by simulation programs, noting that protocols used in these networks are distributed algorithms. The stochastic nature of demands for network resources is modeled in simulation using pseudo-random number generators. The execution of a computer simulation model is comparable to conducting an in-vitro experiment on the target system. All in all, a Monte Carlo simulation program can serve as a flexible testbed for conducting system experimentation without disturbing production networks or constructing software or hardware prototypes.

Several factors contribute to the difficulty of successfully applying simulation technology to performance evaluation. To begin with, the nature of user demands for network resources may be incompletely known or poorly understood. Further, networks are perpetually in a state of flux, because user services and networking technology constantly alter usage patterns. Thus, any model of user traffic would necessarily be approximate or even speculative, especially when new services are under consideration. A review of traffic models will be provided later.

Simulation programs can be used to closely model the processing of user demands for network services. Such models can contain tremendous detail, especially for large networks. However, the execution of detailed models may require prohibitive amounts of computational resources. It is not uncommon for network simulations to require days of processing time on a modern workstation. The analyst must be aware of the trade-off between model detail and simulation execution time. Frugal modeling is often called for, i.e., including only those network functions that have an appreciable impact on the desired performance metrics.

Since executing a simulation is analogous to conducting an experiment involving randomness, simulation output must be treated as random observations. In a similar vein, since a model is viewed as a faithful representation of the target system, instrumentation is required in order to collect statistics and formulate performance predictions. The success of a simulation study hinges upon identifying appropriate performance metrics and then devising a strategy for exploring the ensuing performance response surface. A system response surface is rarely one-dimensional; more often, the desired response of the system is a function of a parameter vector. Correlation in the measured observations must be taken into account when forming statistical performance estimates. For example, positive autocorrelation in a sequence of delay times would manifest itself by bursts of long (short) delays, i.e., if message k has experienced a long (short) delay then it is likely that message $k+1$ will also experience a long (short) delay. The autocorrelated nature of the samples obtained from simulation (or measurement records from operational systems) complicates the task of forming performance predictions. Methods such as replication, batch means, and regeneration are used to account for sample dependence [12]. Furthermore, rare events, such as rejection of user requests for service, communications equipment failures, link cuts, or packet dropping, are performance metrics that characterize networks. Using simulation to estimate the probability of such rare events and their effect on performance is problematic, since vast computational resources may be required to generate a sufficient number of events from which statistical estimates may be formed with adequate statistical confidence.

Even with these difficulties, several factors have combined to increase the use of simulation in network analysis. First, sufficient computational power is available to perform extensive simulation studies of many realistic networks. Second, a new generation of communications network simulation packages has become available. These tools provide an integrated environment to graphically describe the target network, execute the simulation, and analyze the output. Some of these tools provide off-the-shelf models for common networks, obviating the need for the user to write detailed simulation programs. These tools can be directly used by network designers and managers who are not necessarily experts in writing simulation software. Flexible plug-and-play modules are the basis for these tools, the ultimate goal being to make them as easy to use as spreadsheets. However, care must always be exercised in applying simulation, since it is easy to draw erroneous conclusions from inappropriate use of simulation tools. A major goal of this chapter is to provide a guide to the legitimate and productive use of simulation.

Simulation environments can be expected to be put into increasing use, specifically as they become more tightly coupled to other network management tools. For example, several simulation tools accept traces of network data measured by network analyzers (sniffers). These data are used in the simulator as input traffic to drive models of future system configurations. Measured performance data can also be compared to simulated results to tune and validate the models. It is projected that many tools will soon have interfaces to network management systems, e.g., the Simple Network Management Protocol (SNMP). These interfaces will allow a network management tool to automatically configure simulation models based on interactions with the managed system, thereby relieving the user of the sometimes tedious task of describing the base model. Such interfaces can be expected to encourage the integration of simulation into short-term operational management of communications networks.

The purpose of this chapter is to review simulation modeling for network design and management, to describe how traffic and networks may be represented in simulation, and to introduce some criteria for selecting a simulation paradigm for specific applications. The focus of this discussion will be on modeling and statistical aspects of simulation and not on the details of simulation programming.

The rest of this chapter is organized as follows. Section 2 discusses the general structure of network simulation, especially the fundamental concepts underlying discrete event models. Key to the success of performance prediction endeavors is the adequate modeling of random demands for network resources. Section 3 reviews traffic modeling for telecommunications networks. The processing of user requests by the network can be modeled in a variety of ways, e.g., extended queueing networks, Petri Nets, and state transition diagrams; these paradigms, along with the motivation for their evolution, will be discussed in Section 4. Section 5 provides the reader with the basis for evaluating network performance prediction tools. Concepts developed in these sections are then illustrated in Section 6 via a case study. Finally, Section 7 summarizes this chapter with some concluding remarks.

2. COMPUTER SIMULATION OF COMMUNICATIONS NETWORKS

The goal of executing a computer simulation of a communications network is to obtain some notion of how a system would perform under various conditions. From a high-level perspective, telecommunication networks can be thought to consist of users generating demands for network resources and of protocols (distributed algorithms) that control the allocation of network resources to satisfy those demands. Three elements involved in computer simulation of a communications network can thus be identified: 1) modeling the demands for network services and resources; 2) modeling the mechanism the system uses in processing those demands; and 3) instrumenting the model to collect statistics in order to form performance predictions. In this section, each of these elements will be discussed in turn, along with the limitations of the simulation approach.

2.1. Characterizing Demands for Network Resources

Telecommunications applications such as simple phone calls, electronic mail, and file transfers generate demands for network services and resources. These demands can be modeled on several time scales [14], namely, subscription, call, burst, packet, and cell. Analysis is often done at a single time scale, while in some cases, events on a broader time scale need to be considered. However, in each of these instances the resource demands form a stream of requests and resource holding times. For example, consider a LAN-LAN file transfer application operating over a Broadband Integrated Services Digital Network (B-ISDN). The information transport in B-ISDN is based on Asynchronous Transfer Mode (ATM) using small, fixed-length packets called cells, each cell being 53 bytes long. In this case, a request for a file transfer of 1 Mbyte must be broken up into about 122 Ethernet packets, each 1024 bytes in length. Each of these packets is segmented into about 22 ATM cells for transmission. The ensuing 2,684 cells are then transmitted through the B-ISDN fabric. Such file transfer traffic can be modeled as a burst of 122 packets onto an Ethernet and then 2864 cells on the B-ISDN, depending on the level of modeling required and the desired performance metrics. Not only are different time scales involved in modeling demands for network resources, but telecommunications traffic is often time-variant. Typically, there are peak demands in the mid-morning and mid-afternoon.

Fundamentally, the modeler must obtain demand characteristics of the target network and then translate those demands into a traffic model. This is a difficult task, since network traffic is dynamic; it varies broadly with applications and technological evolution. Moreover, a single model would not suffice, as different applications, e.g., e-mail, credit card validation, data entry, and computer-aided design, give rise to very different demand characteristics. These demands would also be a function of the presence of an interactive human interface in the application. Some studies have reported on empirical traffic measurements [15-18], but unfortunately, the measurements could not be mapped to specific applications. For example, measurements of only packet length and interarrival times on a LAN do not characterize end-to-end application-to-application traffic.

In addition to length and interarrival sequences, information on source, destination, and application types is needed for a comprehensive understanding of demand traffic. Time and thought must be invested in studying traffic demands on the system in order to obtain accurate demand models. The quality of the results can be no better than the traffic models used to drive the simulation.

Traffic sources are often characterized by two random processes — one describes the interarrival time process, while the other models the holding time process of resources (service times). For example, in the simple M/M/1 queueing model these are assumed to be Poisson processes, i.e., the interarrival and service times are independent exponential random variables. Unlike the restrictive nature of queueing theory, simulation allows for arbitrary specification of these processes, which are implemented through the use of pseudo-random numbers. (See [12] and Chapter __ for a detailed discussion of pseudo-random number generators.) When developing simulation models, the analyst selects the time scale (e.g., call, burst, or packet) for the model and constructs modules to represent the traffic characteristics. Note that segmentation and reassembly are frequently required. Sequences of pseudo-random numbers are used to generate holding times, message lengths, interarrival times, message/packet errors, and other random phenomena involved in communications networks. Developing accurate models of network traffic is an evolving task, since the rapidly changing nature of communications networks renders them moving targets. Section 3 is devoted to a discussion of traffic modeling.

2.2. Modeling Information Transport Mechanisms in Communications Networks

The main sources of information to be transported over communications networks are human users and software applications. The desire to move this information around creates demands for network services and resources as described in the previous section. The primary tasks of a communications network are to process these demands and to deliver the requisite information to prescribed destinations. Thus, a major element of communications network simulation is the development of a computer program that "mimics" network processing of service demands and resource allocation leading to demand satisfaction.

Networks process demands for communications resources by following appropriate communications protocols. These protocols are executed via algorithms whose lower layers are often implemented in hardware and whose upper layers are coded in software. From a network performance perspective, major functions of the protocols include buffering, error control, flow control, rate control, encapsulation, segmentation and reassembly, routing, contention resolution, blocking, and transmission scheduling. A network simulation imitates the target system by first generating events to represent the arrival of user demands, and then uses a model of the appropriate protocol to imitate the corresponding network processing. Note that the actual protocol is not necessarily implemented; rather, an abstraction of the processing is often used. Usually, only those elements of the protocol that affect the performance metrics of interest are considered.

The generation of user demands and their satisfaction by network protocols are encapsulated in simulation events that are ordered by their time of occurrence. Protocol action depends on the state of the network at the time the demand was issued. For example, a simple routing algorithm may send packets to the output link with the shortest buffer (at that time). This event-based processing falls within a modeling paradigm known as Discrete-Event Simulation (DES) [19]. Most simulation tools for telecommunications networks are based on DES programming. A DES has a simulation clock that changes by random increments. The state of the simulated system is stored in a set of system state variables. Event routines cause state variables to be modified. An event list is used to control the execution sequence of these event routines; the event list consists of simulation events in increasing chronological order. Event routines can add or delete events from the event list, and pseudo-random number generators in the event routines provide the requisite randomness in scheduling future events. From a high-level vantage point, running a simulation is, in essence, the repeated execution of a loop, where at each iteration the most imminent event (the one with the earliest scheduled time) is executed in turn. A flow diagram for DES is shown in Figure 1.

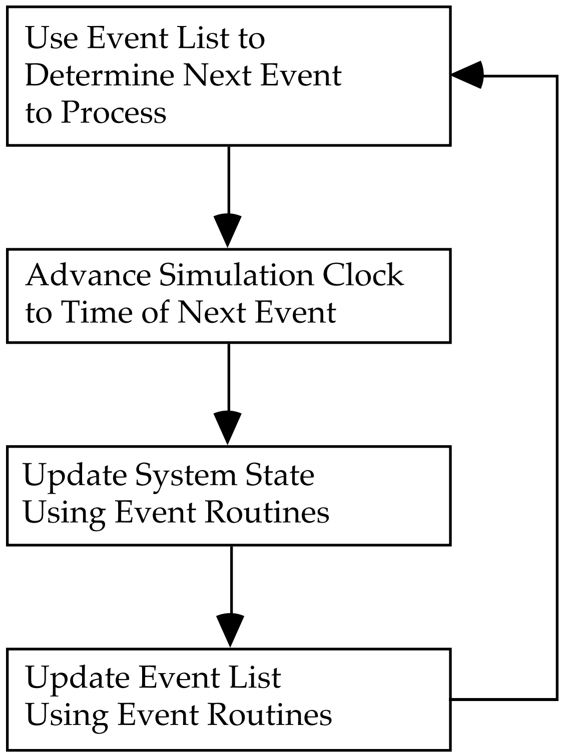


Figure 1: A High-level View of a DES Flow Diagram

Typical events in communications network simulation include the arrival of demands for network resources. Associated with each arrival is a description of its attributes (e.g., demand type, message length) from which the simulation can determine network resources (e.g., processing time, buffer size) needed to satisfy the demand. The time between successive demand arrivals as well as the nature of the required network resources are fundamental elements of traffic modeling, as will be explained later.

The entities acted upon by event routines include calls, messages, packets, and cells. These entities are represented internally by data structures, which are often closely related to the message/packet format as defined by the protocol. For example, source and destination addresses as well as control information and data might be contained in standard packet formats. A data structure representing a packet would contain these elements in addition to simulation-specific information; e.g., a packet-creation time stamp could be included and used for statistics collection. The data field might contain a length indication or, in more detailed models, a pointer to another data structure that represents a network layer packet. An example data structure is shown in Figure 2. Such encapsulation of data structures is a common feature in communications network software; furthermore, the intricacy and complexity of communications objects fit neatly into the object-oriented programming paradigm in a natural way.

Consider messages arriving at a statistical multiplexer as shown in Figure 3. Suppose that a message arrives at time *t* with the state of the system (number of messages at time of arrival) shown as Queue Fill = 0.

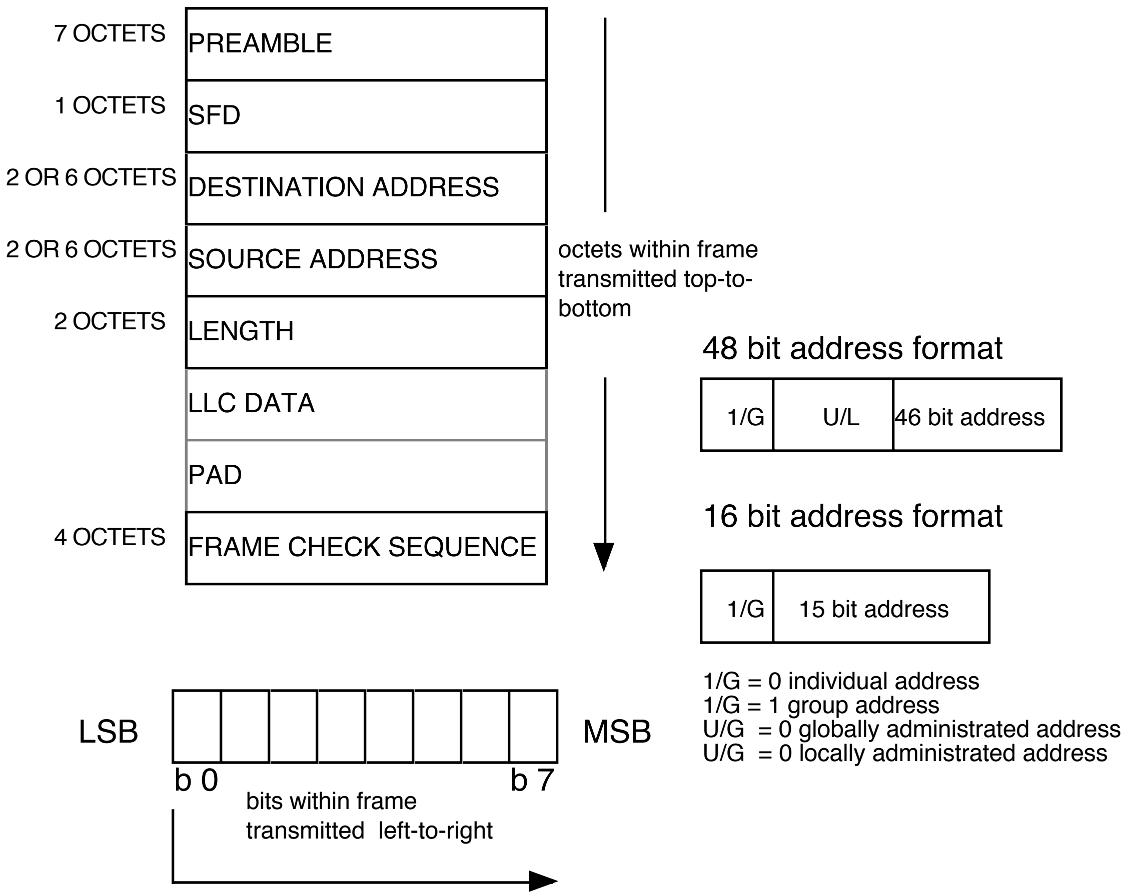


Figure 2: Example of a Communications Packet Data Structure

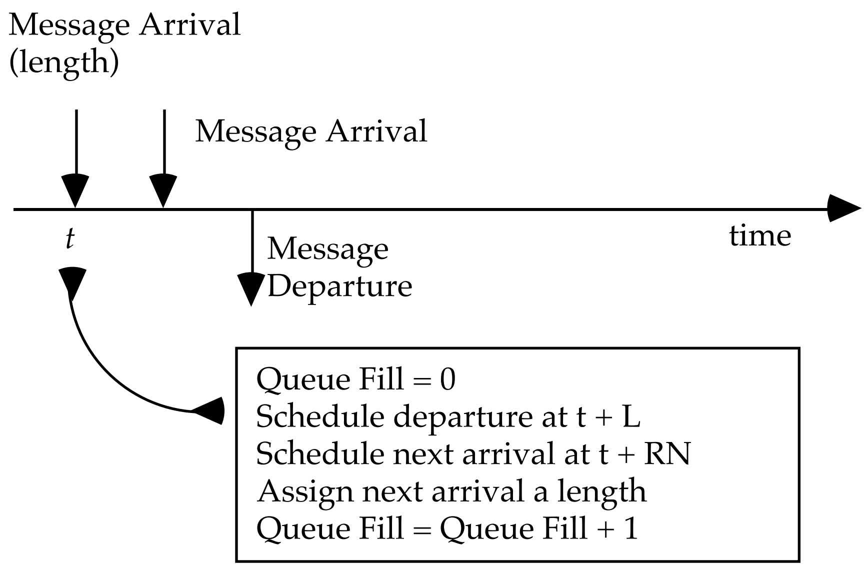


Figure 3: An Example of an Arrival Event in a DES

In this case, the event processing associated with the arrival event takes two actions: 1) it schedules the departure (service completion) of the just-arrived message at time $t+L\tau$, where L is the message length and τ is the unit transmission time; and 2) it schedules the arrival of a new message at time $t+T$, where T is the interarrival time to the next event. The data structure representing an arriving packet might contain its length in bits, bytes, cells, etc., with τ being the corresponding transmission constant. Note that Figure 3 indicates that the system state will be different (non-empty) upon arrival of the second message. In this simple example, the next event to be processed would be the next arrival, followed by the departure of the first arrival. It is clear that a DES must have an initialization mechanism to establish the starting system state and statistical collection routines, in addition to a post-processor for transforming the collected statistics into the desired performance estimates. A coordinating program to control the event list, statistical post-processing, and the initiation and termination of the simulation is also required.

As computing technology has advanced over the past few decades, DES implementation also evolved concomitantly. When digital computers were first applied to simulation of queueing systems, DES was implemented using a general purpose language, such as FORTRAN. Even today, some analysts prefer to develop DES code in a general purpose language, mainly C and C++. Early on, it was recognized that considerable savings in programming effort could be realized through the use of special-purpose modeling languages [20]. The development of textually based simulation languages, such as GPSS [21], SIMULA, SLAM II [22], SIMSCRIPT [23], and MODSIM [24], was primarily motivated by general queueing problems, with the goal of freeing the modeler from low-level coding details. These languages represented the first generation of simulation modeling languages. Further improvements were attained by focusing on a specific class of systems (e.g., communication networks) and adding more simulation services. The second generation of simulation modeling languages was developed with such systems in mind. These included RESQ [1, 23, 25-27], PAWS [28], and Step-1, all of which were still textually based languages. Modern workstations provided enabling technology for the third generation of simulation modeling tools; included in this group are Q+ [29] (formerly PAW), SES/Workbench [30], OPNET [31], Genesis [20], TOPNET [32], NEST [33], and BONeS [34]. A summary of such tools may be found in [35-37]. Unlike their predecessors, these tools are graphically based and provide a visual, interactive environment for performance modeling and evaluation. Their GUI (graphical user interface) supports the construction of network models by manipulating pop-up menus and creating a set of connected icons on a workstation screen. The simulation is executed and controlled interactively, and its output is analyzed in a common environment. Some of these tools can be thought of as graphical simulation programming languages, tailored primarily to communications network modeling. Another feature of some of these tools is their ability to animate the network traffic flows and protocol processing. Such animation is a valuable aid in debugging the models, as well as in communicating the nature of the system to nontechnical personnel. The fourth generation of simulation tools further enhances the previous generation through the addition of interfaces to network monitors and management systems [38]. Modeling system processing of user demands for network services and resources is accomplished by specifying data structures representing the entities flowing through the system, and by coding algorithms that represent the manipulation of those data structures within a DES framework. The underlying DES foundation provided in modern simulation tools allows the user to focus on high-level network modeling constructs (e.g., the process construct) rather than lower-level details (e.g., event scheduling and clock manipulation). Thus, the distinguishing feature of modern simulation languages is their ability to afford the user a high-level hierarchical view of complex systems, often supported by the object-oriented programming paradigm. Various approaches to simulation modeling will be presented later. The issues involved in generating performance predictions from simulation programs will be addressed next.

2.3. Instrumentation of Simulation Models to Collect Statistics and Form Performance Predictions

It is convenient to view a simulation model as a facsimile of the target system, in which instrumentation is needed to collect statistics in order to form performance predictions. Further, executing a simulation is analogous to carrying out an experiment in the lab (in-vitro) — the results of which must be treated as random observations. However, a simulation model provides the opportunity to collect information typically unattainable from the target system; for example, accurate time stamping is easy in a simulation, but often difficult or impossible in real systems. The purpose of this section is to illustrate how to obtain performance predictions from simulation-based experimentation and to identify some potential pitfalls.

To fix the ideas, consider the estimation of message delay. Suppose that the message data structure contains a field for its creation time, i.e., at the instant the message is generated the current simulation time is written into that field. Let $T_c(m)$ be the creation time for the $m-th$ message, and let $T_d(m)$ be its departure time from the system. The corresponding message delay (time in the system) is

$$T_s(m) = T_d(m) - T_c(m).$$

Assume now that in the course of a simulation run, precisely M messages departed from the system. The estimate for the average delay is then given by

$$T_{ave} = \frac{1}{M}\sum_{m=1}^{M} T_s(m).$$

It is often important to estimate the variability of the delay estimator, which is related to the variance of the random variable T_{ave}. Because the samples, $\{T_s(m)\}$, are correlated, direct calculation of the variance must be done with care. That is, using the simple sample variance formula on delay samples may lead to grossly inaccurate estimates. The correlated nature of the samples obtained from simulation runs (or real measurements) complicates the formation of performance predictions. In theory, the variance, $Var[T_{ave}]$, of the random variable T_{ave} can be found from

$$Var[T_{ave}] = \sum_{n=1}^{M}\sum_{j=1}^{M} COV\{T_s(n), T_s(j)\} \Big/ N^2$$

where $COV\{T_s(n), T_s(j)\}$ is the covariance between the $n-th$ and $j-th$ message delays. Estimating the covariance $COV\{T_s(n), T_s(j)\}$ from the sequence, $\{T_s(m)\}$, is difficult unless the sample delays form a stationary sequence. Advanced spectral analysis techniques are sometimes applied to this problem [12]. It is common, however, to take other approaches to deal with the correlation problem. Basically, the desire is to obtain identically distributed, statistically independent (iid) samples of delay. The common approaches to this problem are: 1) simple replication; 2) batch means; and 3) regeneration.

Simple replication entails executing the simulation model K times as illustrated in Figure 4. In this figure $N(t)$ represents the number of packets in the system at time t.

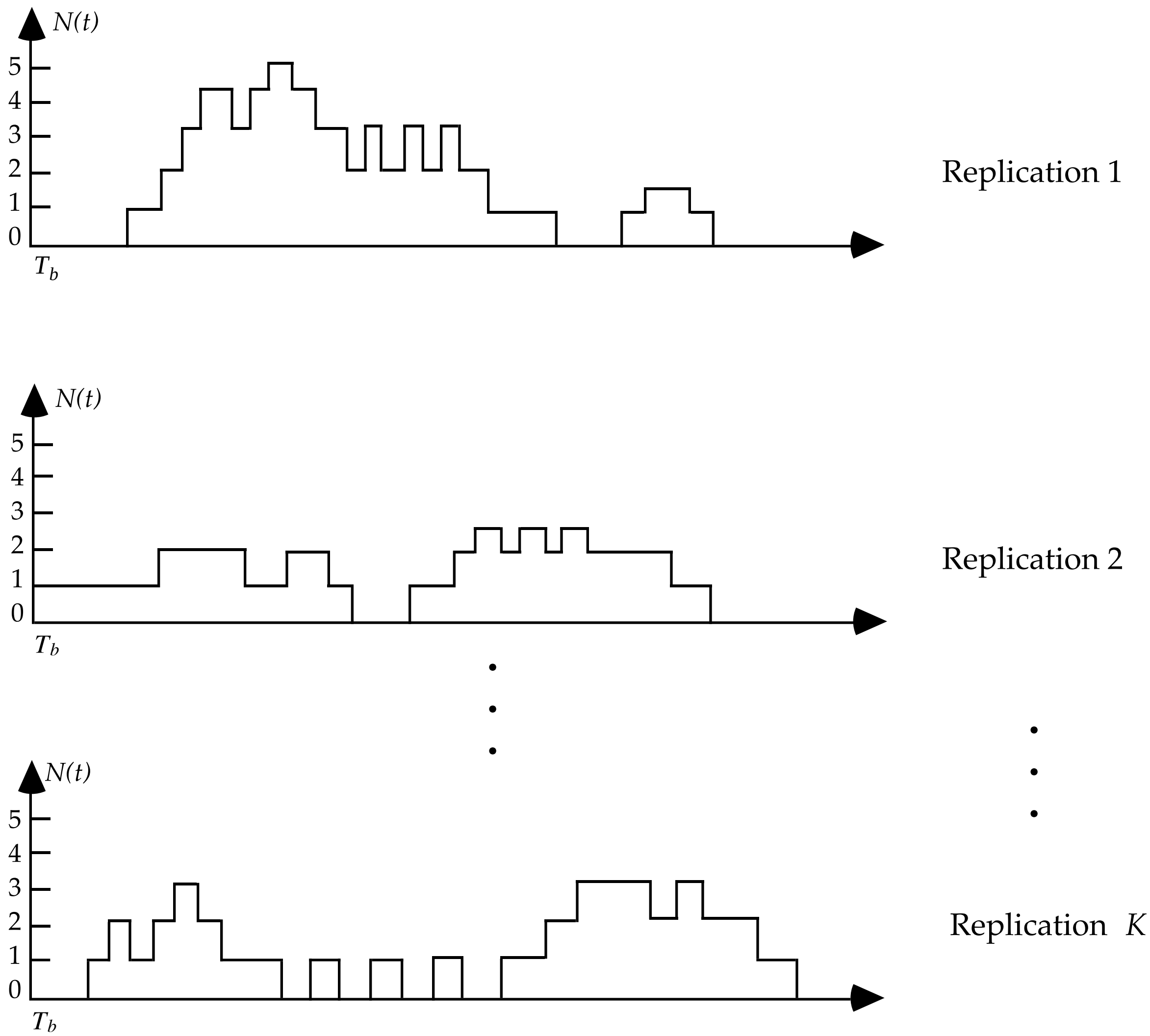


Figure 4: An Example of the Simple Replication Method

Note that the independence assumption means that, in general, each simulation run is started from a different starting point in the pseudo-random sequence, i.e., a different seed is used in the pseudo-random number generators for each run. (Re-executing a simulation model with the same seeds will produce identical results.) The $T_{ave}(k)$, $k=1...K$, obtained from a set of K runs will then be iid. The sample variance, S^2 of T_{ave}, can be directly estimated from the K samples of T_{ave}, namely,

$$\overline{T}=\frac{1}{K}\sum_{j=1}^{K}T_{ave}(j),$$

$$S^2=\frac{1}{K-1}\sum_{k=1}^{K}\left(T_{ave}(k)-\overline{T}\right)^2.$$

Confidence intervals can now be easily formed. For large K, the Central Limit Theorem [39] ensures that the pdf of $\overline{T}$ is approximately normal. Confidence intervals for the expected value of delays are given by

$$\overline{T}\pm z_{1-\alpha/2}\sqrt{\frac{S^2}{K}}$$

for a confidence level of $1-\alpha$, meaning that the probability of the expected value of delay falling between $\overline{T}-z_{1-\alpha/2}\sqrt{\frac{S^2}{K}}$ and $\overline{T}+z_{1-\alpha/2}\sqrt{\frac{S^2}{K}}$ is $1-\alpha$. The parameter $z_{1-\alpha/2}$ is the $1-\frac{\alpha}{2}$ quantile of a unit normal random variate, that is, $P\{N>z_{1-\alpha/2}\}=\frac{\alpha}{2}$ for a normal random variable N with mean zero and variance 1. For example, a 95% confidence interval corresponds to $\alpha=0.05$ and $z_{1-\alpha/2}=1.958$.

Thus, if the confidence intervals are formed 100 times, we would expect the true value of the average delay to fall within these intervals 95 times out of the 100 times. Confidence intervals provide an indication of the quality of the performance estimate; the narrower the confidence interval, the better the estimate. However, there is a trade-off between estimate quality and the cost of the underlying simulation. A smaller confidence interval would be obtained by increasing K, but this, in turn, would increase the execution time of the simulation model.

The batch method divides a sequence of statistical observations from a single simulation run into batches, as shown in Figure 5. Statistics obtained from distinct batches are assumed to be iid, and the above equations for independent samples are applied. Note that the iid assumption will often become increasingly "valid," if some samples in between batches are ignored, i.e., not used in statistical calculations. This has the effect of creating "gaps" between successive batches, which serve to weaken their statistical dependence. In general, observations within and across batches are correlated, but this correlation tends to weaken in most cases, as the separation between distinct batches increases.

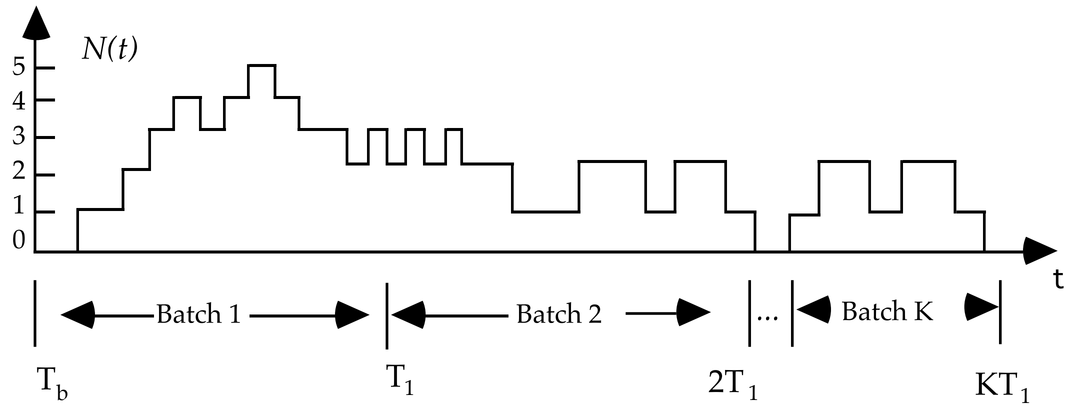


Figure 5: Division of Observations in the Batch Method

Regeneration techniques are appropriate when one can identify time points (called regeneration points) in the simulation that demarcate independent subsequencies (called regenerative cycles). For example, in many queueing systems, the onset of a busy cycle is a regeneration point. Thus, if message m arrives right after the system empties, then its delay will be independent of all previous message delays $1 \ldots m-1$. See [40] for a formal discussion of regenerative processes. Figure 6 illustrates the structure of a regenerative simulation. The salient feature here is that statistics obtained from distinct regenerative cycles are iid. One complication with the regenerative approach is that the time between regeneration points is usually random and may be excessively long. Furthermore, not all systems are regenerative, and even if a system is regenerative, checking for a regeneration condition may be very time-consuming (e.g., checking that a large network is empty, under lightly loaded conditions).

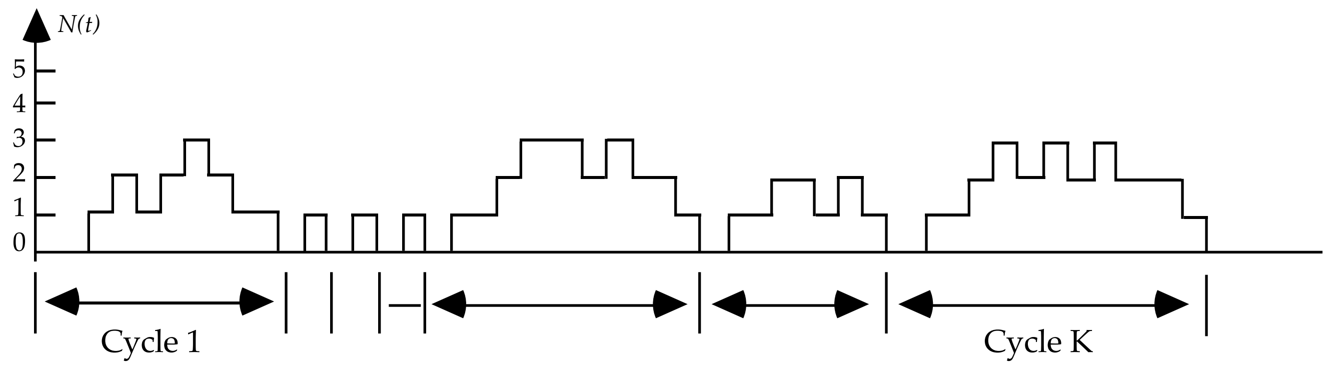


Figure 6: Regenerative Cycles in the Regenerative Method

The above discussion highlights the problems attendant to correlated network measurements. Upon examination of Figures 4, 5, and 6, two other issues come to mind. First, how does the user know when to stop the simulation run? If the simulation is too short, then the associated performance predictions would be unreliable; if it is too long, then computational resources would be wasted. As already noted, simulation can be a computationally intensive task. Stopping rules can be based on the confidence intervals given above or on Tchebycheff's inequality. For example, suppose the analyst wants to execute a sufficient number of replications to ensure that the probability, P_d, that the sample mean falls within a 2Δ interval centered about the true mean, is large, say 0.95. This probability can be bounded using Tchebycheff's inequality, in the following form,

$$Prob\ \left[\overline{T}-\Delta<E[T]<\overline{T}+\Delta\right]>1-\frac{Var\left[\overline{T}\right]}{\Delta^2}\ .$$

The estimate of the variance of $\overline{T}$ is given by S^2/K, where K is the number of replications and S^2 is the sample variance given above. Thus, the desired probability bound can be approximated by

$$Prob\ \left[\overline{T}-\Delta<E[T]<\overline{T}+\Delta\right]>1-\frac{S^2}{K\Delta^2}.$$

The simulation is stopped when $1-\frac{S^2}{K\Delta^2}$ reaches the desired limit [41]. A sequential decision procedure for stopping the simulation would be implemented as follows. Suppose K_1 replications have already been executed, yielding a sample variance of S_1^2. If $1-\frac{S_1^2}{K_1\Delta^2}$ is greater than the desired probability, P_d, then the stopping rule is invoked (sufficient replications have been run); otherwise additional replications are required. The number of replications, K_d, can be estimated by solving $P_d = 1-\frac{S_1^2}{K_d\Delta^2}$ for K_d to yield

$$K_d \geq \frac{S_1^2}{\Delta^2\left(1-P_d\right)}.$$

If the desired confidence has been obtained after completing K_d replications of the simulation, then the process is terminated; otherwise the procedure is continued. The analyst knows that when the algorithm terminates, the expected value of delay is approximately within an interval of width 2Δ about $\overline{T}$, with probability greater than P_d.

In a DES model, as in the real system, the state trajectory of the system evolves from its initial state. For example, if all the buffers are initially empty, then the delay of the first messages entering the network will be lower, on average, than those arriving some time later. A DES is often used to characterize the network in *steady state*, that is, after spending some time in a transient period, the effect of the initial condition wears off and the system reaches a "stable" condition in the sense that its statistics (e.g., time averages) stabilize and change relatively little thereafter. To obtain steady-state performance, the biasing effect of the transient period must be removed or at least sufficiently reduced. Removal of transients typically involves the deletion of statistics generated in the initial period of the simulation, in the sense that these statistics are not used in the calculation of the performance metrics. Determining the end of the transient period (shown as T_b

in Figures 4 and 5) is, in general, difficult. References [12, 42] describe several procedures to identify the end of the transient period. Note that this problem is trivially handled in regenerative simulations, since observations from regenerative cycles are automatically steady-state statistics. In many cases, however, the transient response is the target of the simulation study. For example, studying the response of the network to component failures would call for a transient analysis.

The foregoing discussion shows that care must be taken when constructing performance estimates from simulation output. Correlation, stopping rules, and transient effects must be taken into account. Some network modeling tools aid the user in handling these complex issues; for example, post-processors that automatically apply the batch mean method are already available. The degree to which a modeling environment helps the network designer and manager handle the statistical aspects of simulation output analysis should be carefully considered when selecting a simulation tool.

Traffic modeling is a key element in simulating communication networks, since traffic sources "drive" most communications models. A clear understanding of the nature of the traffic in the target system and the subsequent selection of an appropriate random traffic model are critical to the success of the modeling enterprise. The next section discusses traditional as well as some recently proposed traffic models.

3. TRAFFIC MODELING FOR THE SIMULATION OF COMMUNICATIONS NETWORKS

In this section, we survey the main classes of traffic models commonly used in teletraffic. Such models are used in two fundamental ways: either as part of an analytical model or to drive a discrete-event simulation. The most common modeling context is queueing; traffic is offered to a queue or a network of queues, and various performance measures are calculated.

Simple traffic consists of single arrivals of discrete entities (packets, cells, etc.). It can be mathematically described as a *point process* [39] consisting of a sequence of arrival instants $T_1, T_2, \ldots, T_n, \ldots$. measured from the origin 0; by convention, $T_0 = 0$. Figure 7 shows such a point process measured from LAN traffic [43]. (The arrival indicator at time t in Figure 7 and subsequent figures is used to facilitate the visualization of the arrival process.) There are two additional equivalent descriptions of point processes: *counting* processes and *interarrival time* processes. A counting process $\{N(t)\}_{t=0}^{\infty}$ is a continuous-time, non-negative integer-valued stochastic process, where $N(t) = max\{n : T_n \le t\}$ is the number of (traffic) arrivals in the interval $(0,t]$. An interarrival time process is a real-valued random sequence $\{A_n\}_{n=1}^{\infty}$, where $A_n = T_n - T_{n-1}$ is the length of the time interval separating the $n-th$ arrival from the previous one. The equivalence of these descriptions follows from the equality of events

$$\{N(t)=n\} = \{T_n \le t < T_{n+1}\} = \left\{\sum_{k=1}^{n} A_k \le t < \sum_{k=1}^{n+1} A_k\right\}, \tag{1}$$

since $T_n = \sum_{k=1}^{n} A_k$. We assume throughout that the variance of the A_n is finite.

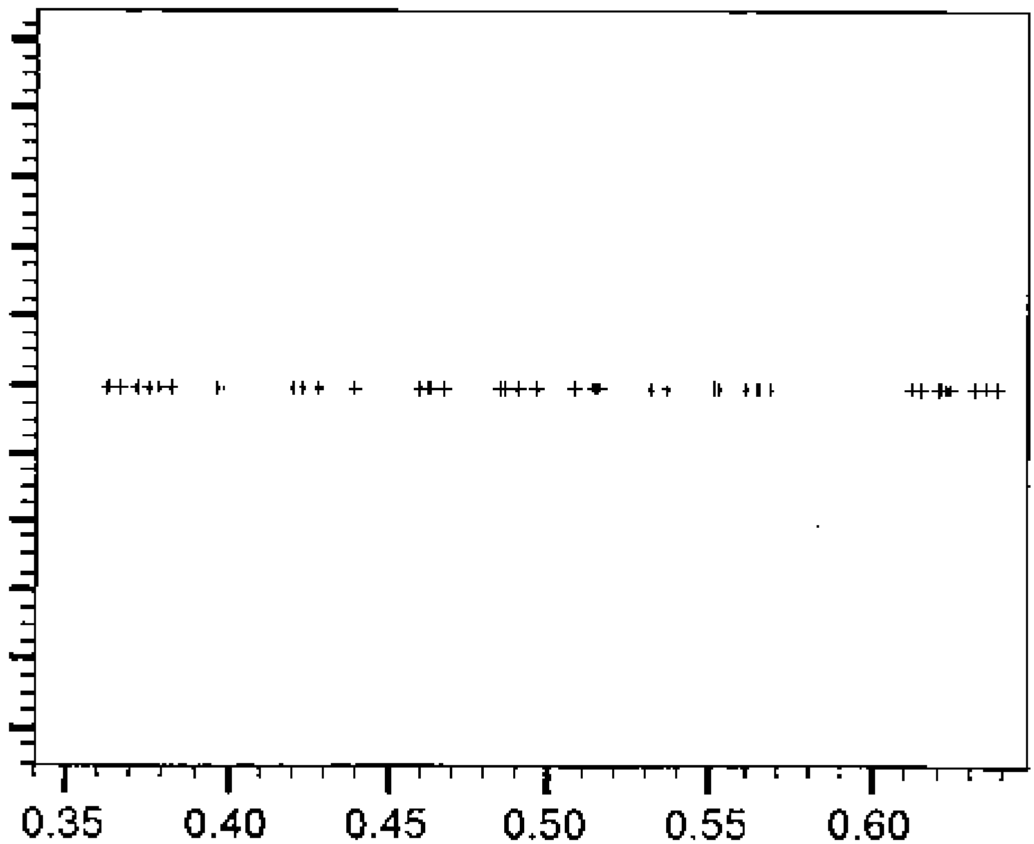


Figure 7: An Empirical LAN Traffic Interarrival Sequence,

(An Example Point Process from [43])

Compound traffic consists of batch arrivals; that is, arrivals may consist of more than one unit at an arrival instant T_n. In order to fully describe compound traffic, one also needs to specify a real-valued random sequence $\{B_n\}_{n=1}^{\infty}$, where B_n is the (random) number of units in the $n-th$ batch. At a higher level of abstraction, B_n may represent some general attributes of the $n-th$ arrival, e.g., the amount of "work" associated with the $n-th$ arrival or its itinerary in a network. Such compound traffic processes, called *marked point processes* [44], are outside the scope of this chapter.

Discrete-time traffic processes correspond to the case when time is slotted as in B-ISDN. Mathematically, this means that the random variables A_n can assume only integer values, or equivalently, that the random variables $N(t)$ are allowed to increase only at integer time instants T_n.

Traffic processes are used to drive simulations in several ways, all of which use one or more pseudo-random number streams to generate sequences of random variables via appropriate transformations. To emphasize this point, we shall use the term *randomly generated* to refer to such computer-generated random sequences, including the deterministic case. In the simplest case (continuous-time or discrete-time simple traffic), a simulation only needs to randomly generate a sequence of interarrival times $\{A_n\}$.

The traffic generation mechanism that would support event algorithms (as discussed previously) is straightforward. Initially, the simulation clock is set to $T_0 = 0$. Next, A_1 is randomly generated and an arrival event is scheduled for time $T_1 = A_1$. This arrival event is placed on the simulation event list or calendar, which consists of chronologically-ordered events. Eventually, that arrival event becomes the most imminent one, the simulation clock is set to A_1, and the next arrival is implemented. Arrival generation proceeds inductively. At simulation time T_n, the $n-th$ arrival event is processed, the next interarrival time, A_{n+1}, is randomly generated, and an arrival event is scheduled for simulation time $T_{n+1}=T_n+A_{n+1}$, and so on. For compound traffic, the simulation randomly generates a batch size B_n (in addition to the interarrival time A_n) and implements the arrival of B_n units at simulation time T_n. Most models call for the sequences $\{A_n\}$ and $\{B_n\}$ to be stochastically independent.

In addition to arrival times and batch sizes, it is often useful (and sometimes essential) to incorporate the notion of *workload* into the traffic description. The workload is a general concept describing the amount of work, W_n, brought to the system by the $n-th$ arriving unit; it is usually assumed independent of interarrival times and batch sizes. A typical example is the sequence of service time requirements of arrivals at a queueing system. In communications networks, service time is the number of transmitted units (bits, packets, cells, etc.) divided by the link rate in the appropriate units. Figure 8 shows an example sequence of message lengths, measured from a LAN. On the other hand, traffic reduces to workload description, when interarrival times are deterministic. A case in point is compressed video, also known as VBR (variable bit rate) video. Here, coded frames (arrivals) have variable and random size (bit rate), and these must be delivered deterministically every 1/30 of a second or so, for high-quality video. The workload consists of coded frame sizes (say, in bits), since frame size is roughly proportional to its transmission time (service requirement). An example of such a workload sequence is shown in Figure 9.

In this section, we describe generic models, which can be used to randomly generate any component of traffic description, be it $\{A_n\}$, $\{B_n\}$ or $\{W_n\}$, but we emphasize simple traffic, described by $\{A_n\}$. We also point out that different traffic streams, corresponding to different telecommunications services (voice, video, file transfer, etc.), can be superposed (multiplexed) to form a realistic heterogeneous mixture of traffic.

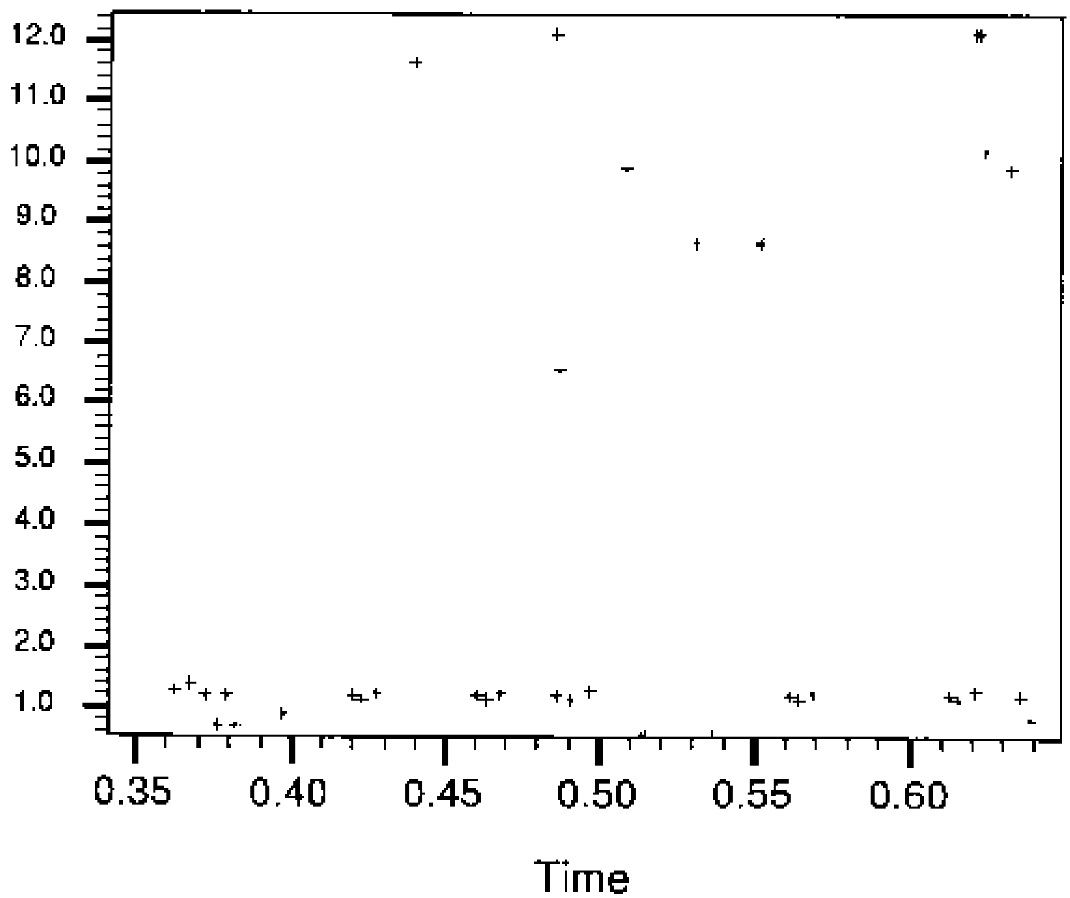


Figure 8. An Example Sequence of Measured LAN Message Lengths

(Data from [43])

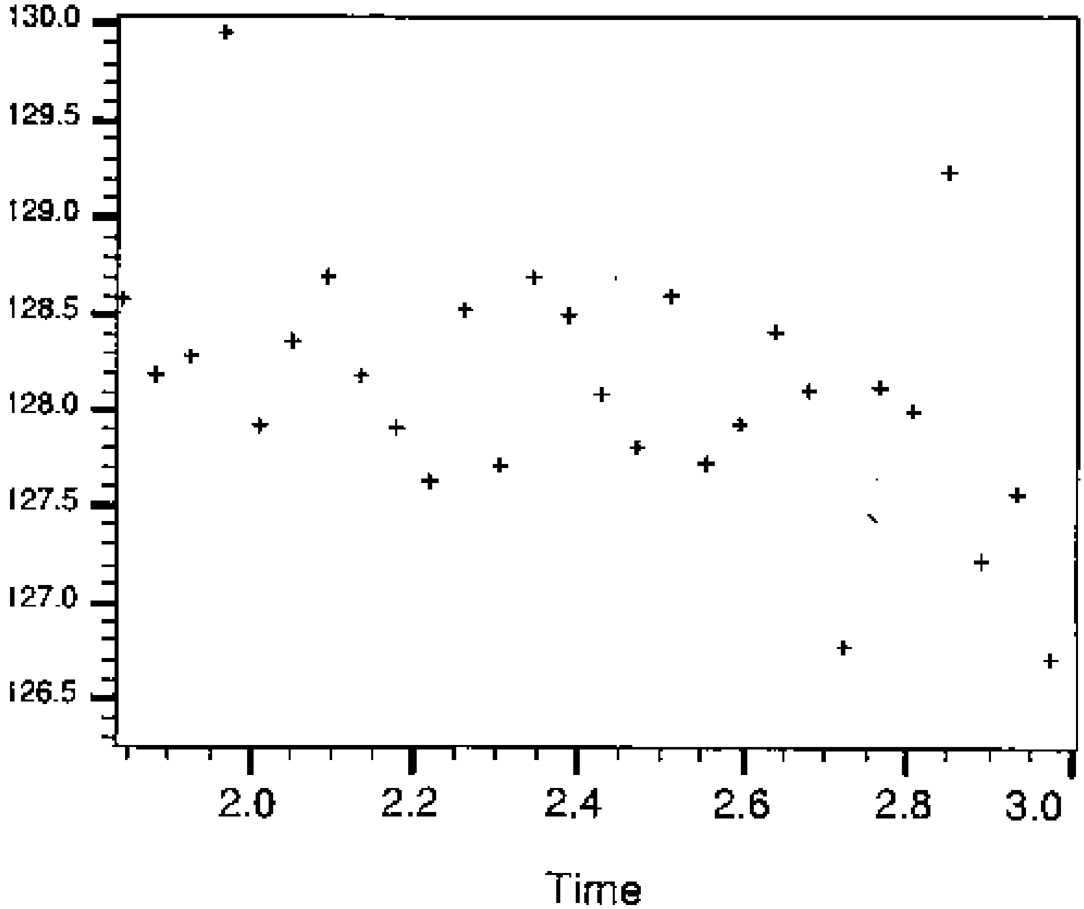


Figure 9. An Example Sequence of Measured Video Workload

(Data from [45])

3.1. Traffic Burstiness

A recurrent theme relating to traffic in communications networks is the traffic "burstiness" exhibited by key services, such as compressed video, file transfer, etc. Intuitively, burstiness is present in a traffic process if the arrival points $\{T_n\}$ appear to form visual clusters; that is, $\{A_n\}$ tends to give rise to runs of several relatively short interarrival times, followed by a relatively long one. The mathematical underpinning of burstiness is more complex. Two main sources of burstiness are due to the shape of the marginal distribution and autocorrelation function of $\{A_n\}$. For example, burstiness would be facilitated by a bimodal marginal distribution of $\{A_n\}$ or by short-term autocorrelations in $\{A_n\}$. Strong positive autocorrelations are a particularly major cause of burstiness. Since there seems to be no single widely accepted notion of burstiness, we shall briefly describe some of the commonly used mathematical measures that attempt to capture it.

The two simplest measures of burstiness only take account of first-order properties of traffic (they are each a function of the marginal distribution only of interarrival times). The first one is the *ratio of peak rate to mean rate* — a very crude measure, which also has the shortcoming of dependence on the interval length utilized for rate measurement. A more elaborate measure of burstiness is the *coefficient of variation*, defined as the ratio of standard deviation to mean, $c_A = \sigma[A_n]/E[A_n]$, of interarrival intervals.

In contrast, the *index of dispersion* measure [46] and the *peakedness* measure [47] do take account of certain aspects of temporal dependence in traffic (second-order properties). For a given time interval of length τ, the index of dispersion for counts (IDC) is the function $I_c(\tau) = Var[N(\tau)]/E[N(\tau)]$, i.e., the variance-to-mean ratio of the number of arrivals in the interval $[0,\tau]$. Since the number of arrivals is related to the sum of interarrival intervals via Eq. (1), the numerator of the IDC includes the autocorrelations of $\{A_n\}$. Similarly, the index of dispersion for intervals is

$$I_n = \frac{var\left[\sum_{j=1}^{n} A_j\right]}{\left(E\left[\sum_{j=1}^{n} A_j\right]\right)^2} :$$

it can be shown that $\lim_{\tau\uparrow\infty} I_c(\tau) = \lim_{n\uparrow\infty} I_c(n)$ [48]. The peakedness concept is related, but more involved. Assume that the traffic stream $\{A_n\}$ is offered to an infinite server group consisting of independent, identical servers with common service time distribution F. Let S be the equilibrium number of busy servers. The peakedness is the functional $z_A[F] = Var[S]/E[S]$, which maps a service time distribution F to a real number. A commonly used peakedness is $z_{exp}(0)$, obtained as a limiting case for an exponential service distribution with service rate approaching 0.

Finally, the *Hurst parameter* can be used as a measure of burstiness via the concept of self-similarity, but the discussion of this notion is deferred until Section 3.8.

3.2. Renewal Traffic Models

This section introduces renewal traffic processes and the important special cases of Poisson and Bernoulli processes. Renewal models have a long history, due to their relative mathematical simplicity. In a renewal traffic process, the A_n are iid (independent, identically distributed), but their distribution is allowed to be general. Unfortunately, with few exceptions, the superposition of independent renewal processes does not yield a renewal process. Those that do, occupy a special position in traffic theory and practice. Historically, queueing models have routinely assumed a renewal offered traffic.

Renewal processes, while simple analytically, have a severe modeling drawback — the autocorrelation function of $\{A_n\}$ vanishes identically for all non-zero lags. The importance of capturing autocorrelations stems from the role of the autocorrelation function as a statistical proxy for temporal dependence in time series. Moreover, positive autocorrelations in $\{A_n\}$ can explain, to a large extent, the phenomenon of traffic burstiness. Bursty traffic is expected to dominate broadband networks, and, when offered to a queueing system, it gives rise to much worse performance (such as mean waiting times) as compared to renewal traffic (which lacks temporal dependence); see [49] for a detailed discussion. Consequently, models that capture the autocorrelated nature of traffic are essential for predicting the performance of emerging broadband networks.

3.2.1 Poisson Processes.

Poisson models are the oldest traffic models, dating back to the advent of telephony and the renowned pioneering telephone engineer A.K. Erlang. A Poisson process [39] can be characterized as a renewal process whose interarrival times $\{A_n\}$ are exponentially distributed with rate parameter λ, that is, $P\{A_n \le t\} = 1 - exp(-\lambda t)$. Equivalently, it is a counting process, satisfying $P\{N(t) = n\} = exp(-\lambda t)(\lambda t)^n / n!$, and the number of arrivals in disjoint intervals is statistically independent (a property known as *independent increments*).

Poisson processes enjoy some elegant analytical properties. First, the superposition of independent Poisson processes results in a new Poisson process whose rate is the sum of the component rates. Second, the independent increment property renders Poisson a memoryless process. This, in turn, greatly simplifies queueing problems involving Poisson arrivals. And third, Poisson processes are fairly common in traffic applications that physically comprise a large number of independent traffic streams, each of which may be quite general. The theoretical basis for this phenomenon is known as Palm's Theorem [50, Vol. II, p. 582]. It roughly states that under suitable but mild regularity conditions, such multiplexed streams approach a Poisson process, as the number of streams grows, but the individual rates decrease so as to keep the aggregate rate constant. Thus, traffic on main communications arteries is commonly believed to follow a Poisson process, as opposed to traffic on upstream tributaries, which is less likely to be Poisson. However, traffic aggregation (multiplexing) need not always result in a Poisson stream; see the discussion in Section 3.8 for a counter-example.

Time-dependent Poisson processes are defined by letting the rate parameter λ depend on time. Compound Poisson processes are defined in the obvious way, by specifying the distribution of the batch size B_n, independent of the A_n.

3.2.2. Bernoulli Processes.

Bernoulli processes are the discrete-time analog of Poisson processes (time-dependent and compound Bernoulli processes are defined in the natural way). Here the probability of an arrival in any time slot is p, independent of any other one. It follows that for interval $(0,k]$, the corresponding number of arrivals is binomial, i.e., $P\{N_k = n\} = \binom{k}{n} p^n (1-p)^{k-n},\ n \le k$. The time between arrivals is geometric with parameter p, i.e., $P\{A_n = j\} = p(1-p)^j, j = 0,1,\ldots$.

3.2.3. Phase-type Renewal Processes.

An important special case of renewal models occurs when the interarrival times are of the so-called *phase type*. Phase-type interarrival times can be modeled as the time to absorption in a continuous-time Markov process $C = \{C(t)\}_{t=0}^{\infty}$ with state space $\{0,1,\ldots,m\}$; here, state 0 is absorbing, all other states are transient, and absorption is guaranteed in a finite time. To determine A_n, start the process C with some initial distribution π. When absorption occurs (i.e., the process enters state 0), stop the process. The elapsed time is A_n, implying that it is a probabilistic mixture of sums of exponentials. Then, restart C with the same initial distribution π, and repeat the procedure independently to get A_{n+1}.

Phase-type renewal processes give rise to relatively tractable traffic models. They also enjoy the property that any interarrival time distribution can be approximated arbitrarily closely by phase-type distributions. Interarrival sequences generated by renewal processes are shown in Figure 10.

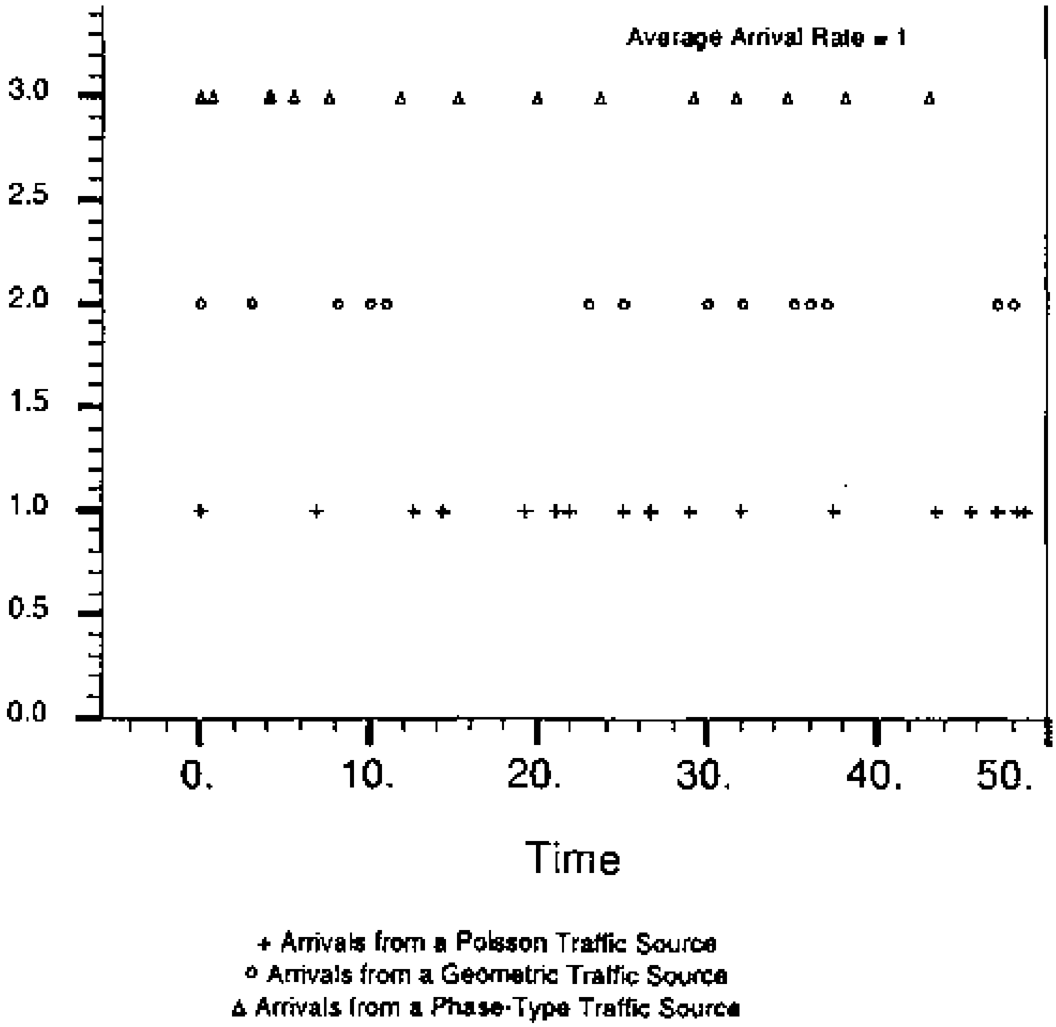


Figure 10: Example Interarrival Sequences from Renewal Processes

3.3. Markov and Markov-Renewal Traffic Models

Unlike renewal traffic models, Markov and Markov-renewal traffic models [39] introduce temporal dependence into the random sequence $\{A_n\}$. Consequently, they can potentially capture traffic burstiness, due to non-zero autocorrelations in $\{A_n\}$.

Consider a continuous-time Markov process $M=\{M(t)\}_{t=0}^{\infty}$ with a discrete state space. In this case, M behaves as follows [39]: it stays in a state i for an exponentially distributed holding time with parameter λ_i, which depends on i alone; it then jumps to state j with probability p_{ij}, such that the matrix $P=[p_{ij}]$ is a probability matrix. In a simple Markov traffic model, each jump of the Markov process is interpreted as signaling an arrival, so interarrival times are exponential, their rate parameter depending on the state from which the jump occurred. This results in dependence among interarrival times as a consequence of the Markov property.

Markov models in slotted time can be defined for the process $\{A_n\}$ in terms of a Markov transition matrix $P=[p_{ij}]$ [39]. Here, state i corresponds to i idle slots separating successive arrivals, and p_{ij} is the probability of a j-slot separation, given that the previous one was an i-slot separation. Arrivals may be single, a batch of units, or a continuous quantity. Batches may themselves be described by a Markov chain, while continuous-state, discrete-time Markov processes can model the (random) workload arriving synchronously at the system. In all cases, the Markov property introduces dependence into interarrival separation, batch sizes, and successive workloads, respectively.

Markov-renewal models are more general than discrete-state Markov processes, yet they retain a measure of simplicity and analytical tractability. A Markov renewal process $R=\{(M_n, \tau_n)\}_{n=0}^{\infty}$ is defined by a Markov chain $\{M_n\}$ and its associated jump times $\{\tau_n\}$, subject to the following constraint: the pair (M_{n+1}, τ_{n+1}) of next state and inter-jump time depends only on the current state M_n, but not on previous states nor on previous inter-jump times. Again, if we interpret jumps (transitions) of $\{M_n\}$ as signaling arrivals, we would have dependence in the arrival process. Also, unlike the Markov process case, the interarrival times can be arbitrarily distributed, and these distributions depend on both states straddling each interarrival interval [39].

MAP (Markovian Arrival Process) is a broad and versatile subclass of Markov renewal traffic processes, enjoying analytical tractability [51]. Here, the interarrival times are phase-type (see above) but with a wrinkle: traffic arrivals still occur at absorption instants of the auxiliary Markov process C, but the latter is not restarted with the same initial distribution; rather, the restart state depends on the previous transient state from which absorption had just occurred. While MAP is analytically simple, it enjoys considerable versatility. Its formulation includes Poisson processes, phase-type renewal processes, and others as special cases [51]. It also has the appealing property that the superposition of independent MAP traffic streams results in a MAP traffic stream governed by a Markov process whose state space is the cross product of the component state spaces.

3.4. Markov-Modulated Traffic Models

Markov-modulated models constitute an extremely important class of traffic models. The idea is to introduce an explicit notion of state into the description of a traffic stream — an auxiliary Markov process is evolving in time, and its current state controls (modulates) the probability law of the traffic mechanism.

Let $M=\{M(t)\}_{t=0}^{\infty}$ be a continuous-time Markov process, with state space $\{1,2,...m\}$ (more complicated state spaces are possible). Now assume that while M is in state k, the probability law of traffic arrivals is completely determined by k, and this holds for every $1\leq k\leq m$. Note that when M undergoes a transition to, say, state j, then a new probability law for arrivals takes effect for the duration of state j, and so on. Thus, the probability law for arrivals is modulated by the state of M (such systems are also called doubly stochastic, but the term "Markov modulation" makes it clearer that the traffic is stochastically subordinated to M).

Certainly, the modulating process can be more complicated than a Markov process (so the holding times need not be restricted to exponential random variables), but such models are far less analytically tractable. For example, Markov renewal processes constitute a natural generalization of Markov-modulated processes with generally distributed interarrival times, but those will not be reviewed here.

3.4.1. Markov-Modulated Poisson Processes.

The most commonly used Markov-modulated model is the MMPP (Markov-Modulated Poisson Process) model, which combines the simplicity of the modulating (Markov) process with that of the modulated (Poisson) process. In this case, the modulation mechanism simply stipulates that in state k of M, arrivals occur according to a Poisson process at rate λ_k. As the state changes, so does the rate.

MMPP models can be used in a number of ways. Consider first a single traffic source with variable rate. A simple traffic model would quantize the rate into a finite number of rates, and each rate would give rise to a state in some Markov modulating process as shown in Figure 11. Certainly, it remains to verify that exponential holding times of rates are an appropriate description, but the Markov transition matrix $Q = \left[Q_{kj}\right]$ of the putative M can be easily estimated from empirical data: simply quantize the empirical data, and then estimate Q_{kj} by calculating the fraction of times that M switched from state k to state j.

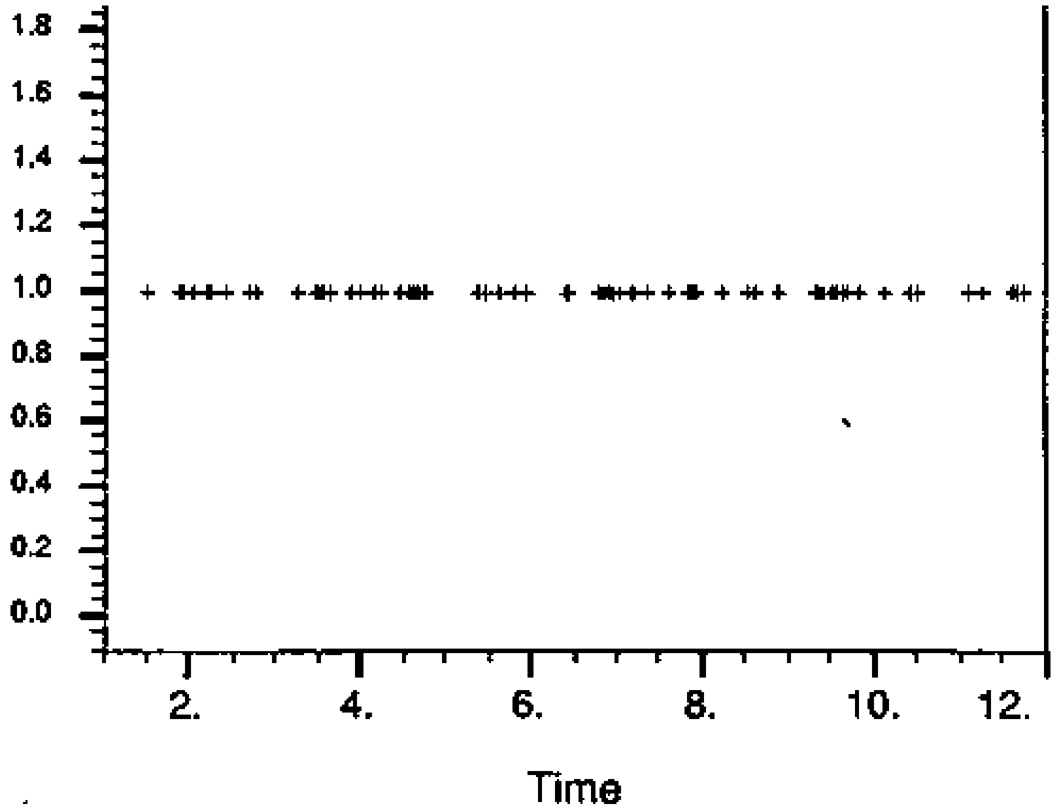


Figure 11: An Example Sample Path from a Three-Rate Continuous-Time Markov-Modulated Process

As a simple example, consider a two-state MMPP model, where one state is an "on" state with an associated positive Poisson rate and the other is an "off" state with associated rate zero (such models are also known as interrupted Poisson for obvious reasons). These models have been widely used to model voice traffic sources [52]; the "on" state corresponds to a talk spurt (when the speaker emits sound), and the "off" state corresponds to a silence (when the speaker pauses for a break). A realization of a discrete-time on/off model is given in Figure 12. This basic MMPP model can be extended to aggregations of independent traffic sources, each of which is an MMPP, modulated by an individual Markov process M_i, as described above. Let $J(t)=\left(J_1(t),J_2(t),...,J_r(t)\right)$, where $J_i(t)$ is the number of active sources of traffic type i, and let $M(t)=\left(M_1(t),M_2(t),...,M_r(t)\right)$ be the corresponding vector-valued Markov process taking values on all r-dimensional vectors with non-negative integer components. The arrival rate of class i traffic in state $\left(j_1,j_2,...,j_r\right)$ of $M(t)$ is $j_i\lambda_i$.

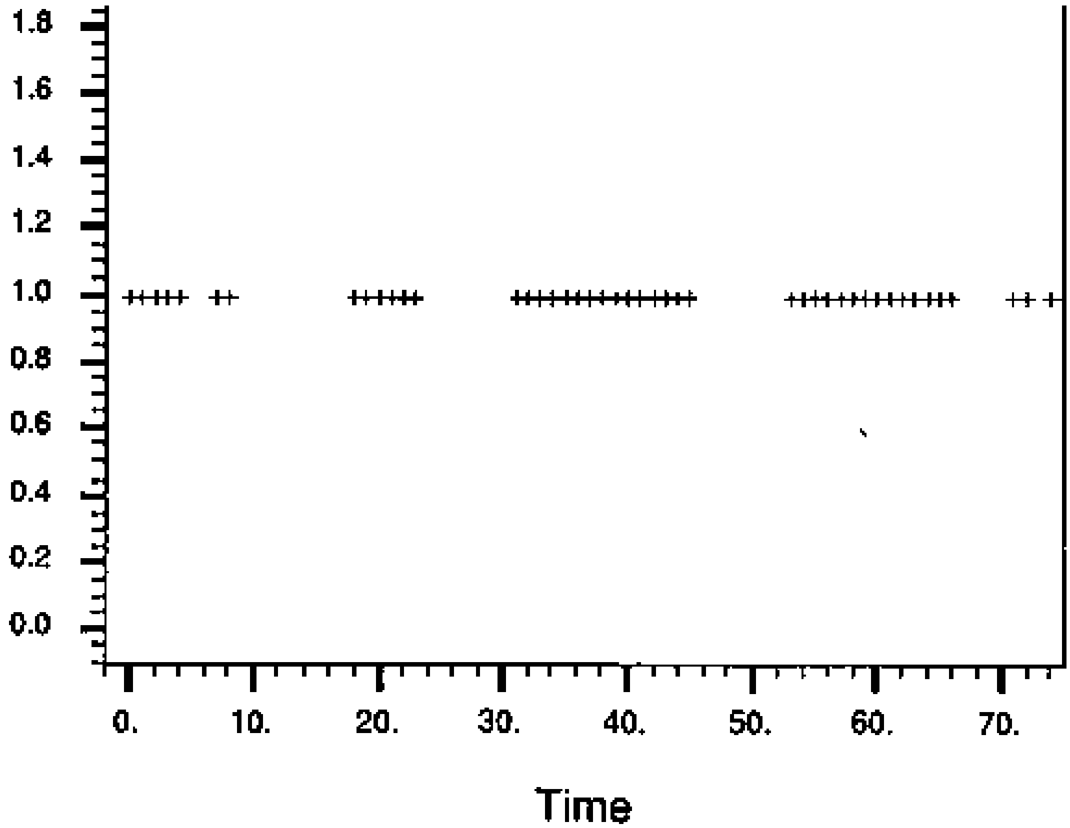


Figure 13: An Example Sample Path of Discrete_Time On/Off Traffic

3.4.2. Transition-Modulated Processes.

Transition-modulated processes are a variation on the state modulation idea. Essentially, the modulating agent is a state transition rather than a state, per se. However, note that a state transition can be described simply by a pair of states whose components are the one before transition and the one after it.

A transition-modulated traffic model in discrete time is described in [53]; its generalization to continuous time is straightforward. Let $M=\{M(t)\}_{t=0}^{\infty}$ be a discrete-time Markov process on the positive integers. State transitions occur on slot boundaries and are governed by an $m \times m$ Markov transition matrix $Q=[Q_{ij}]$. Let B_n denote the number of arrivals in slot n, and assume that the probabilities $P\{B_n=k|M_n=i, M_{n+1}=j\}=t_{ij}(k)$ are independent of any past state information (the parameters $t_{ij}(k)$ are assumed given). Notice that these probabilities are conditioned on transitions (M_n, M_{n+1}) of M from state M_n to state M_{n+1} during slot n. Furthermore, the number of traffic arrivals in slot n is completely determined by the transition of the modulating chain (through the parameters $t_{ij}(k)$).

Markov-modulated traffic models are a special case of Markovian transition-modulated ones: simply take the special case when the conditioning event is $\{M_n=i\}$. That is, $t_{ij}(k)=t_i(k)$ depends only on the state i of the modulating chain in slot n, but it is independent of its state j in the next slot $n+1$. Conversely, Markovian transition-modulated processes can be thought of as Markov-modulated ones but on a larger state space. Indeed, if $\{M_n\}$ is Markov, so is the process $\{(M_n, M_{n+1})\}$ of its transitions.

As before, multiple transition-modulated traffic models can be defined, one for each traffic class of interest. The complete traffic model is obtained as the superposition of the individual traffic models. For queueing studies in discrete time, another wrinkle is the assignment of priorities to different classes, so as to order their arrivals in a buffer [53].

3.5. Fluid Traffic Models

The fluid traffic paradigm dispenses with individual traffic units. Instead, it views traffic as a stream of fluid, characterized by a flow rate (e.g., bits per second), so that traffic count is replaced by traffic volume.

Fluid models are appropriate to cases where individual units are numerous relative to a chosen time scale. Put differently, an individual unit is by itself of vanishingly little significance, just as one molecule more or less in a water pipeline has but an infinitesimal effect on the flow. In the B-ISDN (Broadband Integrated Services Digital Networks) context of ATM (Asynchronous Transfer Mode), all packets are fixed-size cells of relatively short length (53 bytes); in addition, the high transmission speeds (say, on the order of gigabit/second) render the transmission impact of individual cells negligible. The analogy of a cell to a fluid molecule is a plausible one. To further highlight this analogy, contrast an ATM cell with a much bigger transmission unit, say, a compressed high-quality video frame, which consists of on the order of a thousand cells. A traffic arrival stream of coded frames should be modeled as a discrete stream of arrivals, since such frames are typically transmitted at the rate of 30 frames per second. Such a flow rate is shown in Figure 13 for variable bit rate (VBR) video. However, a fluid model is appropriate for the constituent cells.

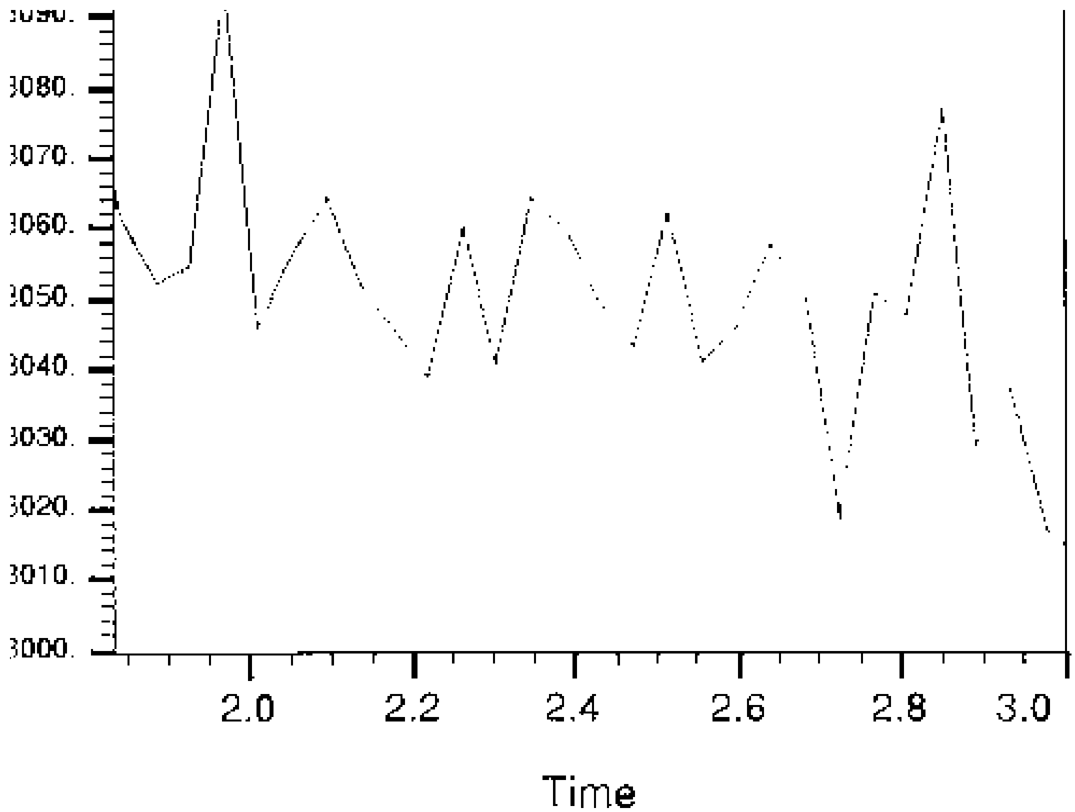


Figure 13: An Example of Empirical VBR Traffic

An important advantage of fluid models is their conceptual simplicity. But important benefits will also accrue to a simulation model of fluid traffic. To see that, consider again a broadband ATM scenario. If one is to distinguish among cells, then each of them would have to count as an event. The time granularity of event processing would be quite fine, and, consequently, processing cell arrivals would consume vast CPU and possibly memory resources, even on simulated time scales of minutes. A statistically meaningful simulation may often be infeasible. In contrast, a fluid simulation would assume that the incoming fluid flow remains (roughly) constant over much longer time periods. Traffic fluctuations are modeled as events signaling a change of flow rate. As these changes can be assumed to happen far less frequently than individual cell arrivals, one can realize enormous savings in computing. In fact, infeasible simulations of cell arrival models can be replaced by feasible simulations of fluid models of comparable accuracy. In a queueing context, it is easy to manipulate fluid buffers. Furthermore, the waiting time concept simply becomes the time it takes to serve (clear) the current buffer. Since fluid models assume a deterministic service rate, these statistics can be readily computed. Typically, though, larger traffic units (say coded frames) are of greater interest than individual cells. Modeling the larger units as discrete traffic and their transport as fluid flow will give us the best of both worlds: we can measure waiting times and enjoy savings on simulation computing resources.

Typical fluid models [54, 55] assume that sources are bursty — of the "on/off" type. While in the "off" state, traffic is switched off, whereas in the "on" state traffic arrives deterministically at a constant rate λ. For analytical tractability, the durations of "on" and "off" periods are often assumed to be exponentially distributed and mutually independent (that is, they form an alternating renewal process). A Markov model of a set of quantized (fluid) traffic rates is presented in [56]. Fluid traffic models of these types can be analyzed as Markov-modulated constant rate traffic. The host of generalizations, described above for MMPP, carries over to fluid models as well, including multiple sources and multiple classes of sources.

3.6. Autoregressive Traffic Models

Autoregressive models define the next random variable in the sequence as an explicit function of previous ones within a time window stretching from the present into the past. Such models are particularly suitable for modeling VBR coded video — a projected major consumer of bandwidth in emerging high-speed communications networks. The nature of video frames is such that successive frames within a video scene vary visually very little (recall that there are 30 frames per second in a high-quality video). Only scene changes (and other visual discontinuities) can cause abrupt changes in frame bit rate. Thus, the sequence of bit rates (frame sizes) comprising a video scene may be modeled by an autoregressive scheme (but see Section 3.6.2 for another modeling approach), while scene changes can be modeled by some modulating mechanism, such as a Markov chain.

3.6.1 Linear Autoregressive Models.

The simplest class of linear autoregressive models [57] has the form

$$X_n = a_0 + \sum_{r=1}^{p} a_r X_{n-r} + \varepsilon_r,\ n > 0,$$

where $X_0,\ldots,X_{p-1}$ are prescribed random variables, the a_r are real constants, and the ε_n are zero-mean, iid random variables, called *residuals*, which are independent of the X_n.

The above equation describes the simplest form of a linear autoregression scheme, called $AR(p)$, where p is the order of the autoregression. In a good model, the residuals ought to be of a smaller magnitude than the X_n, in order to "explain" the empirical data.

The recursive form given above makes it clear how to randomly generate the next random element in the sequence $\{X_n\}_{n=0}^{\infty}$ from previous ones. This simplicity makes them popular candidates for modeling autocorrelated traffic. A simple $AR(2)$ model was used in [58] to model VBR coded video. More elaborate models can be constructed out of $AR(p)$ models combined with other schemes. For example, in [59], video bit rate traffic was modeled as a sum $R_n = X_n + Y_n + K_n C_n$, where the first two terms comprise independent $AR(1)$ schemes and the third term is a product of a simple Markov chain and an independent Normal variate from an iid Normal sequence. The purpose of having two autoregressive schemes is to achieve a better fit of the empirical autocorrelation function; the third term is designed to capture sample path spikes due to video scene changes.

More complicated linear autoregression models, such as ARMA and ARIMA [57], are outside the scope of this chapter. Autoregressive models are typically used to fit the empirical autocorrelation function, but they cannot generally fit the empirical marginal distribution.

3.6.2. TES Models

TES models provide another modeling approach geared toward capturing both marginals and autocorrelations of empirical records [60, 61], traffic included [62]. The empirical TES methodology assumes that some stationary empirical time series (e.g., traffic measurements over time) are available, such as those shown in Figures 7-9. It aims to construct a model satisfying the following three fidelity requirements simultaneously: (1) the model's marginal distribution should match its empirical counterpart (a histogram, in practice); (2) the model's leading autocorrelations should approximate their empirical counterparts up to a reasonable lag; and (3) the sample paths (histories) generated by simulating the model should "resemble" the empirical records. The first two requirements are precise quantitative statements, whereas the third one is a heuristic qualitative one. Nevertheless, it is worth adopting this third subjective requirement, keeping its interpretation at the intuitive level; after all, common sense tells us that if a model gave rise to time series that are entirely divorced in "appearance" from observed ones, then this would weaken our confidence in the model, and vice versa.

TES processes come in two flavors: TES^{+} and TES^{-}. The superscript (plus or minus) is a mnemonic reminder of the fact that they give rise to TES processes with positive and negative lag-1 autocorrelations, respectively. TES models consist of two stochastic processes in lockstep, called *background* and *foreground* sequences, respectively. Background TES sequences have the form

$$U_n^{+} = \begin{cases} U_0, & n=0 \\ \langle U_{n-1}^{+} + V_n \rangle, & n>0 \end{cases} \quad \text{or} \quad U_n^{-} = \begin{cases} U_n^{+}, & n \quad even \\ 1-U_n^{+}, & n \quad odd \end{cases}$$

Here, U_0 is distributed uniformly on $[0,1)$; $\{V_n\}_{n=1}^{\infty}$ is a sequence of iid random variables, independent of U_0, called the *innovation sequence*, and angular brackets denote the modulo-1 (fractional part) operator $\langle x\rangle = x - max\{integer\ n:\ n \le x\}$. Background sequences play an auxiliary role. The real targets are foreground sequences of the form

$$X_n^+ = D(U_n^+), \quad \text{or} \quad X_n^- = D(U_n^-),$$

where D is a transformation from $[0,1)$ to the reals, called a *distortion*.

It can be shown that all background sequences are Markovian stationary, but while the transition structure of $\{U_n^+\}$ is time-invariant, that of $\{U_n^-\}$ is time-dependent. The marginal distributions of both $\{U_n^+\}$ and $\{U_n^-\}$ are uniform on $[0,1)$, *regardless* of the probability law of the innovations $\{V_n\}$ [60]. The *inversion method* [63] allows us to transform any background uniform variate to a foreground variate with an arbitrary marginal distribution. To illustrate this idea, suppose one has an empirical time series $\{Y_n\}_{n=0}^{N}$, from which one computes an empirical density $\hat{h}_Y$ and its associated distribution function $\hat{H}_Y$. Then, the random variable $X = \hat{H}_Y^{-1}(U)$ has density $\hat{h}_Y$. Thus, TES foreground sequences can match any empirical distribution, through a choice of a suitable distortion.

The empirical TES methodology actually employs a composite two-stage distortion of the form

$$D_{Y,\xi}(x) = \hat{H}_Y^{-1}(S_\xi(x)),\ x \in [0,1),$$

where $\hat{H}_Y^{-1}$ is the inverse of the empirical histogram distribution and S_ξ is a "smoothing" operation, called a *stitching transformation*, parameterized by $0<\xi<1$ and given by

$$S_\xi(y)=\begin{cases} y/\xi, & 0\le y<\xi \\ (1-y)/(1-\xi), & \xi\le y<1 \end{cases}$$

For $0<\xi<1$, the effect of S_ξ is to render the sample paths of background TES sequences more "continuous-looking." Because stitching transformations preserve uniformity, the inversion method via $\hat{H}_Y^{-1}$ guarantees that the corresponding foreground sequence will have the prescribed marginal distribution $\hat{H}_Y$. The empirical TES modeling methodology takes advantage of this fact, which effectively decouples the fitting requirements of the empirical distribution and the empirical autocorrelation function. Since the former is automatically guaranteed by TES, one can concentrate on fitting the latter. This is carried out by a heuristic search for a pair (ξ, f_V), where ξ is a stitching parameter and f_V is an innovation density; the search is declared a success on finding that the corresponding TES sequence gives rise to an autocorrelation function that adequately approximates its empirical counterpart and whose simulated sample paths bear "adequate resemblance" to their empirical counterparts.

In practice, efficient searches of this kind must rely on software support. TEStool is a visual interactive software environment designed to support TES modeling [64]. TEStool allows the user to read in empirical sample paths and calculate their empirical statistics (histogram, autocorrelation function, and spectral density) in textual and graphical forms. It further provides services to generate and modify TES models and to superimpose the corresponding TES statistics on their empirical counterparts. The search proceeds in an interactive style, guided by visual feedback: each model modification triggers a recalculation and redisplay of the results. TES model autocorrelations and spectral densities are calculated numerically from fast and accurate formulas developed in [60, 61]. This activity is further simplified by restricting the innovation densities f_V to be step functions. Simple densities like that can be readily modified graphically, since steps are visually represented by rectangles, and those can be created, deleted, stretched, and moved easily with the mouse.

Figure 14 displays a typical TEStool screen at the end of a TEStool modeling session based on an empirical record of a (random) sequence of encoded (compressed) video frames [65]. Video information is rarely transmitted over a network in its raw form. Rather, engineers take advantage of the considerable visual redundancy inherent in digitized pictures to compress each frame into a fraction of its original size. The compressed frames have random sizes (bit rates) that are then transported over the network and decoded at their destinations. The term VBR (variable bit rate) video is used to refer to this kind of video traffic.

The specific data modeled in Figure 14 were a random sequence of frame bit rates, generated by a VBR video sequence of a football scene whose frames were compressed by a variant of DCT (discrete cosine transform) [66]; see also [67] for a review of compressed video modeling using the TES methodology. The screen is sub-divided into four canvas areas designed to display various types of graphics; the graphical user interface is controlled by buttons and menu selections. Each statistical canvas superimposes an empirical statistic with its TES-model counterpart; a legend at the top of these graphs identifies the constituent curves.

The upper-left canvas contains the empirical and model sample paths, the latter being generated by a Monte Carlo simulation; the upper-right canvas contains the corresponding histograms; the lower-left canvas contains the empirical autocorrelation function and its numerically computed model counterpart; the lower-right canvas contains a joint specification of a TES sign, stitching parameter, and an innovation density.

Notice that the empirical time series in Figure 14 (upper-left canvas) exhibits very high short-term autocorrelations (lower-left canvas), attesting to the bursty nature of this type of encoded video. A simple renewal model would not be appropriate, since it cannot capture burstiness due to temporal dependence. In contrast, the TES model displayed in Figure 14 exhibits generally good statistical agreement with the empirical data, in accord with the three fidelity requirements stipulated above. Such source models can be used to generate synthetic streams of realistic traffic to drive simulations of communications networks.

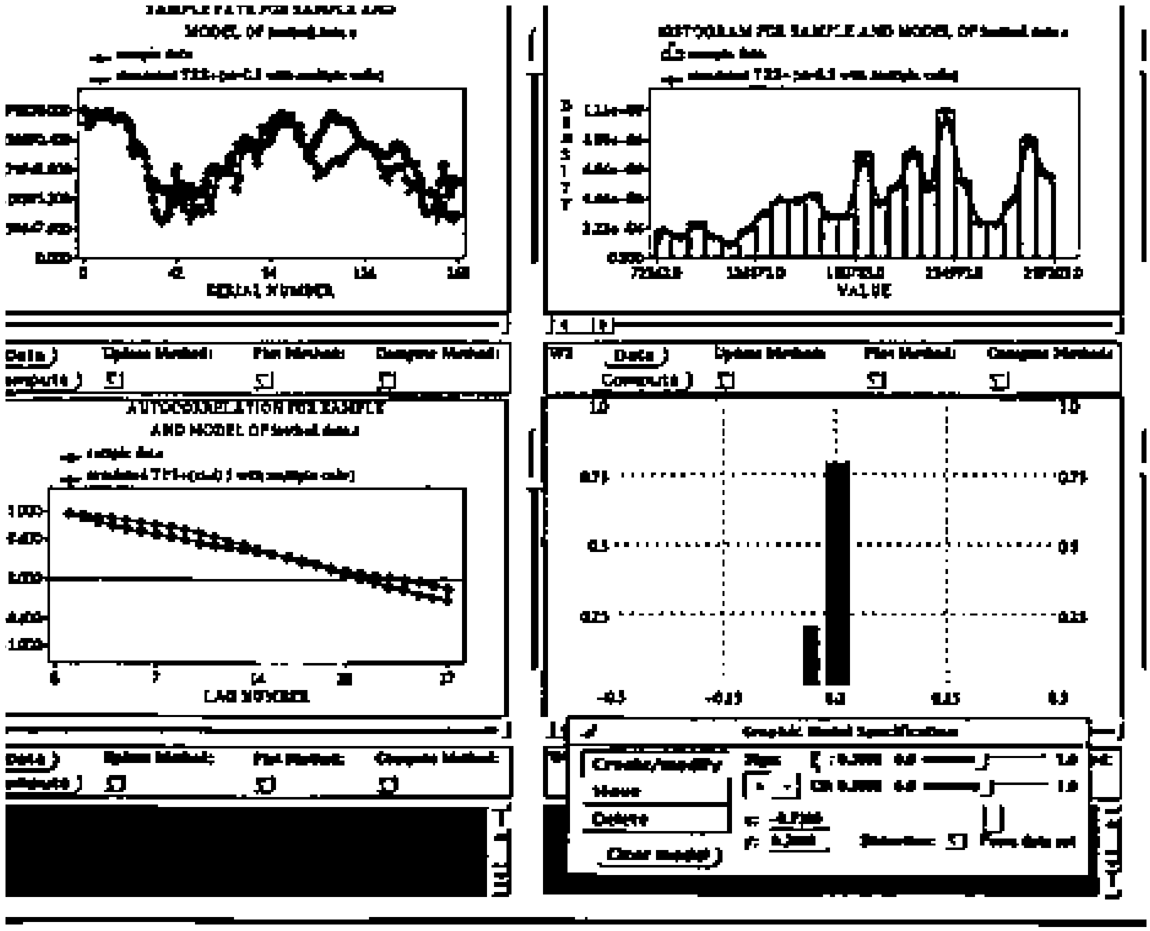


Figure 14: TES Model of DCT-Compressed VBR Video

3.7. The Frequency Domain Approach

The Frequency Domain Approach (FDA) focuses on second-order statistics of offered traffic and their effect on queue response to that traffic [68-70]; it has been motivated by the need to characterize multimedia traffic in high-speed networks. FDA is distinguished by the fact that it directly utilizes the frequency domain (the traffic spectral functions) and advocates their use as a unified traffic measurement for analyzing and controlling queueing systems with heterogeneous offered traffic.

Analogously to periodic input functions in signal processing, [68] uses elements of DC, sinusoidal, rectangular pulse, triangle pulse, and their superpositions to represent various second-order traffic statistics and to observe their effect on queueing performance. The main finding is that only the low frequencies in the traffic spectrum have a significant effect on queueing statistics. However, this approach has limited applications, since it does not capture the stochastic aspects of a prescribed traffic process. To this end, [70] uses multi-state MMPP traffic to characterize binary sources (on/off traffic) and to study the effect of their second-order statistical properties on queue length and loss rate statistics.

[69] introduces a modeling technique for constructing Markovian traffic processes that match a prescribed power spectrum. Let Q be the infinitesimal generator (transition rate matrix) of an N-state Markov chain, which modulates the Poisson rate of the traffic process with rate vector $\bar{\gamma}$. It is shown that the eigenstructure of Q characterizes the effect of such traffic on queueing performance. Assume that Q is diagonalizable with eigenvalues $\bar{\lambda}=(\lambda_0,\lambda_1, \dots ,\lambda_{N-1})$. The power spectrum of the random (Poisson) rate process can then be expressed by

$$P(\omega)=\sum_{j=0}^{N-1}\frac{-2\psi_j\lambda_j}{\lambda_j^2+\omega^2}\,. \tag{1}$$

Essentially, each eigenvalue, λ_j, contributes a bell-shaped component in the traffic rate power spectrum. Each bell is described by the average power, ψ_j, the central frequency, $Im[\lambda_j]$, and the half-power bandwidth, $-2Re[\lambda_j]$. Letting $\vec{\psi}=(\psi_0,\psi_1,\ldots,\psi_{N-1})$, the traffic power spectrum is characterized by the pair $(\vec{\psi},\vec{\lambda})$, and the goal of FDA is to construct $(Q,\vec{\gamma})$ from $(\vec{\psi},\vec{\lambda})$ — a rather difficult task, involving the so-called inverse eigenvalue problem. The technique in [69] uses a special class of Markov chains, called *circulants*, with infinitesimal generators of the form

$$Q=\begin{pmatrix} a_0 & a_1 & a_2 & \cdots & a_{N-1} \\ a_{N-1} & a_0 & a_1 & \cdots & a_{N-2} \\ a_{N-2} & a_{N-1} & a_0 & \cdots & a_{N-3} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_1 & a_2 & a_3 & \cdots & a_0 \end{pmatrix} \tag{2}$$

where each row is a circular right-shift of the previous one. A circulant-type modulated traffic rate process is then characterized by the pair $(\vec{a},\vec{\gamma})$, $\vec{a}$ being the first row of Q. Its power spectrum can be shown to have the representation

$$\vec{\lambda}=\sqrt{N}\vec{a}F^*,\qquad \vec{\psi}=\frac{1}{N}\left|\vec{\gamma}F^*\right|^2 \tag{3}$$

where F is a Fourier matrix with $F_{kj}=\frac{1}{\sqrt{N}}e^{-\frac{2\pi kj}{N}i}$ and $i=\sqrt{-1}$. Then $F^{-1}=F^*$

where F^* is the conjugate transpose of F. More importantly, one can deduce the corresponding pair, $\left(\vec{a},\vec{\lambda}\right)$, that gives rise to a prescribed power spectrum, characterized by a pair, $\left(\vec{\lambda},\vec{\psi}\right)$, by taking the following constrained inverse discrete Fourier transform,

$$\begin{aligned} \vec{a} &= \frac{1}{\sqrt{N}}\vec{\lambda}F \quad \textit{subject to } a_j \geq 0 \textit{ for } j>0 \textit{ and } a_0 = -\sum_{j=1}^{N-1} a_j \ , \\ \vec{\gamma} &= \sqrt{N}\vec{\beta}F \quad \textit{subject to } \gamma_j \geq 0,\ \forall_j, \textit{ with } \beta_j = \sqrt{\psi_j e^{i\theta_j}} \ , \end{aligned} \tag{4}$$

where $\vec{\theta} = \left(\theta_0, \theta_1, \ \ldots \ , \theta_{N-1}\right)$ is a phase vector. Hence, $\vec{a}$ depends only on $\vec{\lambda}$, and $\vec{\gamma}$ only depends on $\left(\vec{\psi},\vec{\theta}\right)$. It should be noted that while $\vec{\theta}$ is unrelated to the power spectrum, it influences the steady-state distribution and higher-order traffic statistics [69]. The numerical analyses in [69, 71] are based on the Folding Algorithm, which is a fast computational method for analysis of finite quasi-birth-death process with level-dependent transitions [72].

A significant advantage of using circulants in FDA is the ability to identify the impact of first-order (marginal distribution), second-order (power spectrum or autocorrelation function), and higher-order traffic statistics (such as the bispectrum and trispectrum) on queue length and loss rate statistics. Interestingly, only second-order traffic statistics appear to have a significant effect on queueing performance, with the low frequencies exerting the most effect. This observation has important potential applications to traffic measurement, especially since first-order and second-order statistics are much easier to measure than higher-order ones. FDA has been applied, in this vein, to traffic rate control [73], to link capacity allocation and network control [74], and to the design and analysis of buffer congestion control [71].

3.8. Self-Similar Traffic Models

Recent studies of high-quality high-resolution traffic measurements have revealed a new phenomenon with potentially important ramifications to the modeling, design, and control of broadband networks. These include an analysis of hundreds of millions of observed packets over an Ethernet LAN in an R&D environment [43] and an analysis of a few millions of observed frame data generated by VBR video services [45] (see Figures 7 and 9). What emerged from these studies is that packet traffic appears to be statistically *self-similar* in the sense of [75]. Intuitively, a self-similar (or fractal) phenomenon exhibits structural similarities across all (or at least a wide range) of time scales. In the case of packet traffic, self-similarity is manifested in the absence of a natural length of a "burst": at every time scale ranging from a few milliseconds to minutes and hours, similar-looking traffic bursts are evident. Self-similar stochastic models include *fractional Gaussian noise* [76] and *fractional ARIMA processes* [77].

Self-similarity manifests itself in a variety of different ways: a spectral density that diverges at the origin $\left(1/f^{\alpha}-noise,\ 0<\alpha<1\right)$, a non-summable autocorrelation function (indicating long-range dependence), and a variance of the sample mean that decreases (as a function of the sample size n) more slowly than $1/n$; see [78] for details. The key parameter characterizing these phenomena is the so-called *Hurst parameter*, H, designed to capture the degree of self-similarity in a given empirical record [79] as follows. Let $\{Y_k\}_{k=1}^{n}$ be an empirical time series with sample mean $\overline{Y}(n)$ and sample variance $S^2(n)$. The rescaled adjusted range, or R/S statistic, is given by $R(n)/S(n)$ with

$R(n) = max\left\{\sum_{i=1}^{k}\left(Y_i - \overline{Y}(n)\right): 1 \leq k \leq n\right\} - min\left\{\sum_{i=1}^{k}\left(Y_i - \overline{Y}(n)\right): 1 \leq k \leq n\right\}$. It has been found empirically that many naturally occurring time series appear to obey the relation

$$E[R(n)/S(n)] \cong n^H$$

for large n, with a Hurst parameter H typically about 0.73. On the other hand, for renewal and Markovian sequences, it can be shown [76] that the above equation holds with $H = 0.5$ for large n. This discrepancy, generally referred to as the *Hurst phenomenon*, is a measure of the degree of self-similarity in time series and can be estimated from empirical data [45].

From a mathematical point of view, self-similar traffic differs from other traffic models in the following way [78]. Let m be a time unit representing a time scale, such as $m = 10^s$ seconds $(s = 0, \pm 1, \pm 2, \ldots)$. For every time scale m, let $X^{(m)} = \left\{X_n^{(m)}\right\}$ denote the time series computed as the number of packets (bytes, cells, etc.) per time unit m in the traffic stream. Traditional traffic models have the property that, as m increases, the "aggregated" processes $X^{(m)}$ tend to a sequence of iid random variables (covariance stationary white noise). On the other hand, the corresponding aggregation procedure of empirical traffic data yields time series $X^{(m)}$, which reveal two related types of behavior, when plotted against time. They either appear visually indistinguishable from one another ("exactly self-similar") but distinctively different from pure noise, or they converge to a time series with a non-degenerate autocorrelation structure ("asymptotically self-similar"). In contrast, simulations of traditional traffic models rapidly converge to white noise after increasing the time scale by about 2 or 3 orders of magnitude. Similarly, when trying to fit traditional traffic models to self-similar traffic data, the number of parameters required typically grows as the sample size increases. In contrast, self-similar traffic models are able to capture the observed fractal nature of packet traffic in a *parsimonious* manner (i.e., with about 1-4 parameters). Parameter estimation techniques are available for many self-similar models, as well as Monte Carlo methods for generating long traces of synthetic traffic [45].

Potential implications of self-similar traffic on issues related to design, control, and performance of high-speed, cell-based networks are currently under study. For example, it can be shown that many of the commonly used measures for burstiness (see Section 3.1) do not characterize self-similar traffic [43]. Contrary to commonly held beliefs that multiplexing traffic streams tends to produce smoothed-out aggregate traffic with reduced burstiness, [43] shows that aggregating self-similar traffic streams can actually intensify self-similarity (burstiness) rather than diminish it. From a practical vantage point, there are also indications that traffic congestion in self-similar networks may have broadly differing characteristics from those produced by traditional traffic models. Comparisons of queueing performance under self-similar versus traditional traffic scenarios are currently being studied.

3.9. Summary of Traffic Modeling

This section has dealt with statistical modeling of demands for network resources. Realistic traffic models are essential for the analysis of network performance. Traditionally, such models are based on observed traffic patterns and not on any analysis of the interaction between the communicating entities. For example, consider a simple file transfer protocol. A few user keyboard strokes might well trigger a large traffic burst. Capturing such correlation is difficult but can be accomplished through detailed pattern analysis of network traffic or detailed simulation of applications. Unfortunately, there is a dearth of such models, and detailed application simulation is computationally intensive. Thus, the random process models discussed in this section will continue to play a dominant role in network performance prediction.

The next section presents various methods of representing networks, given the elements of network simulation, traffic modeling, protocol processing, and performance estimation.

4. REPRESENTATION PARADIGMS FOR COMMUNICATIONS NETWORKS

A variety of software tools is currently available to provide a modeling and analysis environment for DES models of communications networks. Each of these tools contains a system configurator, simulator or analytical back end, post-processor, and model library [80]. A system configurator is used to describe a network architecture and node interactions. Execution control of the simulation model is handled by the simulator, while analytical calculations are handled by an appropriate back end. This may be a complex task involving the simultaneous specification of multiple cases or distributing the computational tasks over a network of processors. As previously explained, the post-processor aids the user in analyzing the results, often via extensive statistical and graphical displays. Furthermore, there are many common communications network components (e.g., window flow control procedures) that are required in many different models. Model libraries are often available to facilitate model component reuse.

The main difference among these tools stems from their paradigms or "world views," i.e., their approach to representing the network in the system configurator. This section describes some of the desired features of an environment for DES modeling of communications networks with special emphasis on the system configurator. It will then proceed to introduce some of the common representation paradigms for communications networks.

4.1. Desired Features of Software Modeling Environments.

A major function of a modeling environment is to specify the network model under study for the purpose of statistical analysis. We now describe the set of features that facilitate that function.

- Graphical Modeling

A salient feature of most approaches to network modeling is their graphical orientation. The advent of graphical workstations has provided the requisite computing power, allowing the user to describe the network model by means of graphical representations. A network is represented by icons that are typically interconnected, and the associated topology depicts the flow of information through the network. A major goal of a visual representation is to hide immaterial detail, both in model abstraction and DES implementation. An important advantage of graphically-oriented computer modeling is that it makes the model easier to understand and manipulate, reduces the need to write code, and encourages model reuse. The graphical interface provides a mechanism for describing systems in a way that more closely matches the user's concept of the system operation, as compared with textual programming languages. Iconized graphs provide an efficient means for grasping the topology of complex networks. Modeling becomes accessible to more users, since a user need not be an expert in a specific programming or simulation language to efficiently use graphically-based tools. Graphical orientation tends to minimize debugging, since it is relatively easy to use debugged off-the-shelf models from libraries and conceptually easier to trace node interactions. When new models are needed, the focus is on debugging the logical and modeling errors instead of annoying violations of language syntax. Furthermore, the graphical orientation lends itself to animation, i.e. the visualization of information flow through the network. Animation further eases the debugging process and provides a mental image of system operation, which allows the analyst to obtain a qualitative feel for the system under study by observing its evolution in time. Such intuition is much harder to acquire from traditional, textually-oriented models.

- Event and Protocol Processing

Like any DES, the modeling environment should provide simulation services (clock, event list, etc.) as well as the mechanism to manage the asynchronous nature of communications networks (queueing data structures and operations are a notable example). Randomness and asynchronous processing (e.g., interarrival times and message lengths in a queueing context) are some of the elements that lead to the asynchronous nature of communications networks. The extended queueing network paradigm specifically aims to capture these phenomena, as will be described later. As a general guideline, the user should avoid general-purpose languages lacking built-in or libraried DES services and should select visual, interactive environments over purely textual ones.

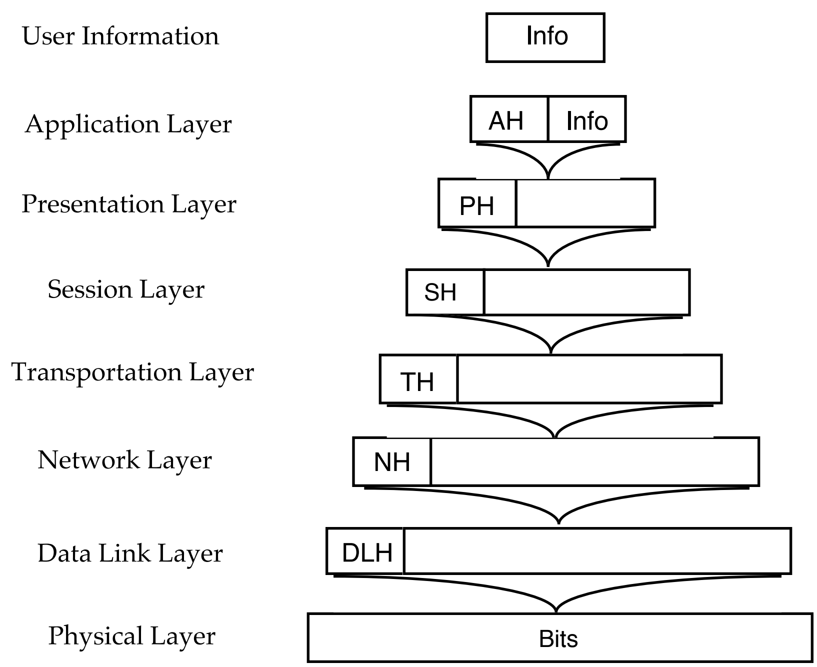
The performance of a communications network is a function of protocols, that is, the algorithms that define how each network element responds to communications events. Graph-based models such as Petri nets and state transition diagrams are often used to model protocol aspects of communication networks [31, 32, 81, 82].

Note that to provide an appropriate modeling environment, both the extended queueing network and graph-based paradigms had to be modified to accommodate the full range of functions needed to model both asynchronous events and protocols.

- Hierarchy

Modern communication networks are both extremely complex and geographically distributed. A major problem in network management, as well as modeling, is handling this complexity. A structural hierarchy is needed to manage the complexity of network topology, protocols, and data structures. A hierarchical framework simplifies the description of network elements and their interactions by allowing system elements to be constructed either from primitives or from other elements. At each layer in the hierarchy, the user deals with a level of detail that is appropriate for that layer. Details of the layer below and the one above are largely hidden. Communication networks are particularly well suited for hierarchical modeling. A hierarchy also provides a convenient way to describe the geographical extent of a communications network. Switching nodes ("offices") in telephone networks are often described by a "telephone office" hierarchy.

A good example of a hierarchical specification is afforded by the layering concepts of the Open Systems Interconnection (OSI) layered architecture [2], which aims to minimize design complexity. Each layer in the architecture performs specific functions and provides services to adjacent layers; the details of *how* the services are performed are hidden from the adjacent layers. Both the interfaces between layers and the associated data structures must be well defined, though particular implementations may vary. As a consequence of such a layered architecture, data structures used at higher layers are encapsulated into the data structures of lower layers (see Figure 15). Note that the header information at each layer provides just enough information to process the packet at that layer. In a similar vein, a modeling environment should accommodate layered models through the hierarchy of data structures and operations that allow encapsulation, where each layer provides specific functions via a fixed interface. The ensuing modularity would facilitate substitution of layer algorithms to study specific implementations and encourages software reuse.



AH	Applicaton Header	TH	Transport Header
PH	Presentation Header	NH	Network Header
SH	Session Header	DLH	Data Link Header

Figure 15: Data Structure Encapsulation in the OSI Layered Architecture

- User Interface

The next set of features pertain to an intelligent user interface and performance analysis mechanism. A graphical user interface (GUI) allows the user to construct models by interconnecting icons or blocks, often in a hierarchical manner. The modeling environment should monitor model construction and employ "expert system" techniques to perform extensive consistency checking, the goal being to identify errors as early as possible. This would reduce the time of programming cycles, consisting of model building followed by software error and the attendant debugging. Consistency checks serve to detect inadvertent or non-obvious errors and increase user confidence in model correctness. Additionally, the environment should facilitate teamwork (group modeling). A good user interface should also provide on-line documentation and help as well as integration facilities for melding components into new models. Extensive statistical services should be provided to support output analysis (formation of performance estimates for the system under study). A flexible mechanism for the instrumentation (collection of performance data from the models) is essential. Typically, a simulation model is executed repeatedly with various parameter settings, or even with system modifications, as model evolution tracks design evolution. The modeling environment should facilitate comparisons of results from several simulation studies and system design alternatives; overall, it should provide a convenient mechanism for performing design iterations.

It is important that all aspects of modeling, analysis, and design mentioned above be handled within a unified environment. The user interface of a framework should have a modular structure containing a model library, graphical editors, simulation and database managers, and a post-processor.

4.2. Common Paradigms for Communications Network Modeling

Construction of simulation models of communications networks is a particularly time-consuming and error-prone process when general purpose languages, such as C or FORTRAN, are used. Even when special-purpose simulation languages are employed, the development of simulation models is difficult. Also, reuse and extendibility of such models is problematic. Graphical approaches to the simulation of communication networks have been developed to mitigate these difficulties. The "world views" of the main approaches will be briefly explained next.

- The Extended Queueing Network (EQN) Paradigm

Several graphical approaches have been based on the EQN paradigm, e.g., Q+ (formerly PAW) [29], GENESIS [20], SES/Workbench [30], and RESQ [25]. In this paradigm, communications links, buffers, disks, and other elements are modeled as resources. A set of entities, called transactions (e.g. packets, messages, etc.), flow among the resources and contend for their use [82]. EQN models contain a broader set of network elements, as compared with traditional analytical queueing models, admitting the modeling of a wide class of systems. Specifically, standard queueing networks do not model important system characteristics, such as simultaneous resource possession, parallelism, and synchronization. Standard queueing networks, which are composed of sources, queues, and sinks, must be augmented. In the EQN paradigm, system modeling employs nodes, transactions, and rules to control their flow. This paradigm is particularly well suited for capturing the asynchronous and queueing aspects of communications systems. Further, it is often possible to obtain analytic approximations directly from extended queueing network models (see [29]). However, modeling detailed protocol functions is often awkward under this paradigm.

- The Petri Network Paradigm

Petri nets were originally developed to study the interconnection properties of concurrent and parallel activities [83], but they have also found extensive applications in modeling communications network protocols. The elements of Petri nets include tokens, places, triggers (transitions), and arcs. Places contain tokens, whereas arcs connect places to triggers. A trigger is enabled if each of its input places contains one or more tokens. Upon firing, a token is removed from each of its input places and a token is added to each of its output places. Petri nets are particularly well suited for the study of protocol correctness, such as absence of deadlock. To illustrate, [84] describes how Petri nets were used to specify and prove deadlock-free progress for protocol correctness. This paradigm has been modified to capture the temporal aspects of communication networks, particularly queueing models. These are sometimes known as Timed Petri Nets. Stochastic activity networks are a combination of Petri nets and queueing models [85-87]. A modification of Petri nets for network simulation is given in [81]. Another paradigm of modified Petri net modeling is called a PROT net [88]; a simulation tool, called TOPNET, has been developed [32] to support PROT-based modeling.

- The Finite-State Machine (FSM) Paradigm

FSM models provide a popular mechanism for protocol representation. An FSM consists of a set of states, input, outputs, and transition rules [83]. Upon receiving an input while in a given state, the FSM will change state (based on a state transition function) and produce an output. FSMs are graphically represented by state transition diagrams — an advantage of FSM-based tools. Such diagrams are ubiquitous in communications standards documents. However, simple FSMs must be augmented so as to model the queueing and temporal aspects of communications networks. OPNET [31] is a tool based on the FSM paradigm.

- The Data-Flow Block-Diagram Paradigm

The data-flow block-diagram paradigm is based on the observation that descriptions of communications protocols and networks require characterizations of data structures (e.g. packets) and operations on these structures performed by the network and its protocol. The latter are described by a directed graph of functional blocks, which transform and possibly queue their input data structures to produce output data structures.

BONeS [89] is a tool based on the data-flow block-diagram paradigm. Here, blocks are constructed in a tree hierarchy; a tree's leaves consist of basic operations called *primitives*. Paradigm elements consist of data structures, arcs, ports, and blocks. Data structures contain information to be transported from one block to another. Typical data structures contain information encapsulated within packets with the possible addition of modeling-specific information, e.g., packet length or time stamps. These data structures are organized in a hierarchy and can follow or mimic standard protocol packet formats. Block-connecting arcs represent paths along which "data structures" flow within the protocol. Such arcs are connected to blocks via ports. A complete network is specified via a set of nested, hierarchical block diagrams. Block diagrams may be used to construct new blocks from pre-existing ones, and these can be used, in turn, to construct still higher-level blocks, and so on. These features provide a mechanism for modeling both the synchronous (protocol processing) and asynchronous (event driven) aspects of communications networks.

Each network modeling paradigm has a specific natural domain of applicability, i.e., system classes to which a given paradigm is particularly well matched. For example, extended queueing networks are excellent for modeling store-and-forward networks. Petri nets and FSMs are well suited for studying protocol correctness. The block-diagram approach can be tailored to have the characteristics of FSM, Petri nets, and EQN [34]. Application of a paradigm outside its natural domain may be awkward. From a network analysis perspective, care should be taken to select the paradigm that best handles the specific system and performance issues under study. The next section presents some criteria for selecting a modeling environment.

5. CRITERIA FOR SELECTING A COMMUNICATION NETWORK MODELING ENVIRONMENT

The selection of a performance modeling tool should take into account many factors, including technical, economic and organizational considerations. The relative importance of these factors may depend on the user and the particular modeling project under study. Following is a list of some important considerations.

- Cost

Performance prediction tools for communication networks range in cost from several hundred dollars to tens of thousands of dollars. This represents a broad range of tool capabilities.

- Availability

Some tools are commercial products, while others are proprietary or available only for research purposes.

- Training required

The level of specialized training required to master these tools varies greatly. The requisite training depends on the complexities of the underlying paradigm and the abstraction required under the modeling paradigm; ease of use and generality tend to be inversely related. The modeling paradigms described above require different levels of modeling abstraction, e.g., the formulation of a network operation in terms of a Petri net model may be a nontrivial task requiring a high degree of network and modeling expertise. Thus, on-line help offered by tools should also be considered.

- Documentation

Related to training are the level and quality of the documentation provided with the environment. On-line documentation and its ease of use should be examined. The ease of documenting user-developed models should also be considered.

- Flexibility

As pointed out, a major motivation for using simulation is its ability to track an evolving target network and adapt to new and evolving technology. Thus, the tool must be open-ended in its ability to support new and evolving functionalities.

- Model Reuse

Tools should also be evaluated on the ease of reusing previously developed network models or components thereof. The ability to exchange simulation models among modeling groups is a factor to be considered.

- User Interface

Ease of use is a fundamental consideration. The interface should aid the user in constructing the model, and obvious inconsistencies and semantic errors should be automatically identified, preferably as the model is being constructed.

- Model Debugging

The modeling environment should contain good debugging facilities (e.g., animation) to assist the user in identifying and correcting logical errors.

- Portability

With the rapid increase in computational power and the proliferation of hardware, it is important that the tools be portable across platforms. This portability could be facilitated, for example, by using standard window systems.

- Computational Efficiency

Simulation is a computationally-intensive task. The implementation of the simulation engine within the modeling environment affects its execution speed and could spell the difference between a feasible simulation and an infeasible one. Efficiency is, therefore, an important consideration.

- Harmonious Operation in a Network Environment

The computing platform for modeling tools often consists of a network of workstations, possibly with components from different vendors. A desirable attribute of a tool is its ability to take advantage of a networked computing environment, e.g., the ability to designate the workstation where the simulation code is to be executed.

- Experimentation Manager

Simulation studies may involve extensive experimentation for purposes of comparison and optimization. An experimentation manager can help organize the experiments, run them, and store the results.

- Post-Processor

A post-processor can greatly facilitate output analysis, by supporting statistical analysis of results and aiding in their interpretation. An example is the ease of forming confidence intervals using any of the schemes described in Section 2. Furthermore, a post-processor should support iteration over system parameters to easily form performance curves.

- Animation

Animation is a powerful tool for visualizing the operation of the systems as well as debugging simulation models. A good animation facility is highly desirable.

- Usage

Consideration should be given to the nature of the modeling project. A tool designed to provide life cycle support for a network will most likely differ markedly from a tool suitable for a quick, one-time modeling effort.

- Interface with the External World

Simulation tools become all the more valuable if they include interfaces to the external world, e.g., the target network. The extent of such interfaces should be a major factor in the evaluation process for network managers.

The evaluation of specific modeling environments is a multi-criteria decision based on the factors enumerated above. In addition, one should consider the relative merits of general-purpose and special-purpose languages as compared with the graphically-oriented modeling environments. Such a comparison is summarized in Tables 1-3.

Table 1: Relative Merits of General-Purpose Languages

Advantages	**Disadvantages**
• Wide availability	• Longer programming and debugging time
• Few restrictions imposed on the model	• Difficult verification
• User may have prior knowledge of the language	• Unless object-oriented, limited ability to reuse models
• Generally more computationally efficient	• Model enhancement and evolution are difficult

Table 2: Relative Merits of Special-Purpose Languages

Advantages	**Disadvantages**
• Provide built-in simulation services to reduce programming effort	• Must adhere to a particular “world view” of the language
• Provide error-checking techniques superior to those provided in general-purpose languages	• Availability and support
• Provide a brief, direct vehicle for expressing the concepts arising in a simulation study	• Cost
• Provide ability to construct user subroutines required as a part of any simulation routine	• Increased computer running time
• Contain set of subroutines for common random numbers	• Training required to learn the language and modeling paradigm
• Facilitate collection and display of data produced.	
• Facilitate model reuse.	

Table 3: Relative Merits of Computer-Aided Analysis and Design Environments

Advantages	**Disadvantages**
• Provide a complete integrated performance analysis environment	• Tailored to a specific modeling paradigm
• Graphically-based	• May be tied to a specific hardware platform
• Typically integrate language, database, prior knowledge, and statistical analysis packages	• Increased execution time
• Support management of models and input/output data	• Cost
• Facilitate model reuse and group model development	

Procuring a performance modeling tool may involve a considerable financial outlay. Furthermore, as with any software, porting costs are quite high, so any acquisition entails at least a project-long commitment to the tool with the usual consequences thereof. Thus, careful analysis of tool capabilities versus perceived needs should be carried out to match the right tool to a given modeling undertaking.

6. A CASE STUDY: SENSITIVITY ANALYSIS

As communications needs continue to grow on both academic and business campuses, network managers are faced with rapidly evolving systems that must satisfy the expanding needs of their user communities. Simulation provides an appropriate tool for addressing performance issues associated with communications networks. As an illustration, a large campus network study will be presented. Here, protocol detail is of secondary interest, while system architecture and topology are the primary focus. A model will be constructed by reusing existing component (subsystem) models, and the performance issues will be addressed. This example is drawn from [89], where the BONeS performance tool was utilized.

The purpose of this section is to illustrate the use of simulation in analyzing a substantial network. The main concern is the performance evaluation of alternative network designs, given common network elements. (See [89] for the details of the individual component models). The topology of the campus network under consideration is given in Figure 16. This system consists of an FDDI backbone, which serves as a high-speed data trunk connecting the smaller

LANs.

Connected to the network are a number of workstations capable of exchanging (sending and receiving) data packets. The packet arrival process at each workstation is assumed to be Poisson. Routing is random, i.e., each workstation, with the exception of the workstation sending the packet, is equally likely to be chosen as the destination for each new packet. Without knowledge of specific traffic patterns, these are idealized assumptions made for the sake of this example; in realistic networks, the arrival process is often not Poisson and the traffic pattern may have “hot spots” rather than being uniformly distributed across all possible destinations. The particular traffic model selected is based on empirical data of transaction demands for network resources.

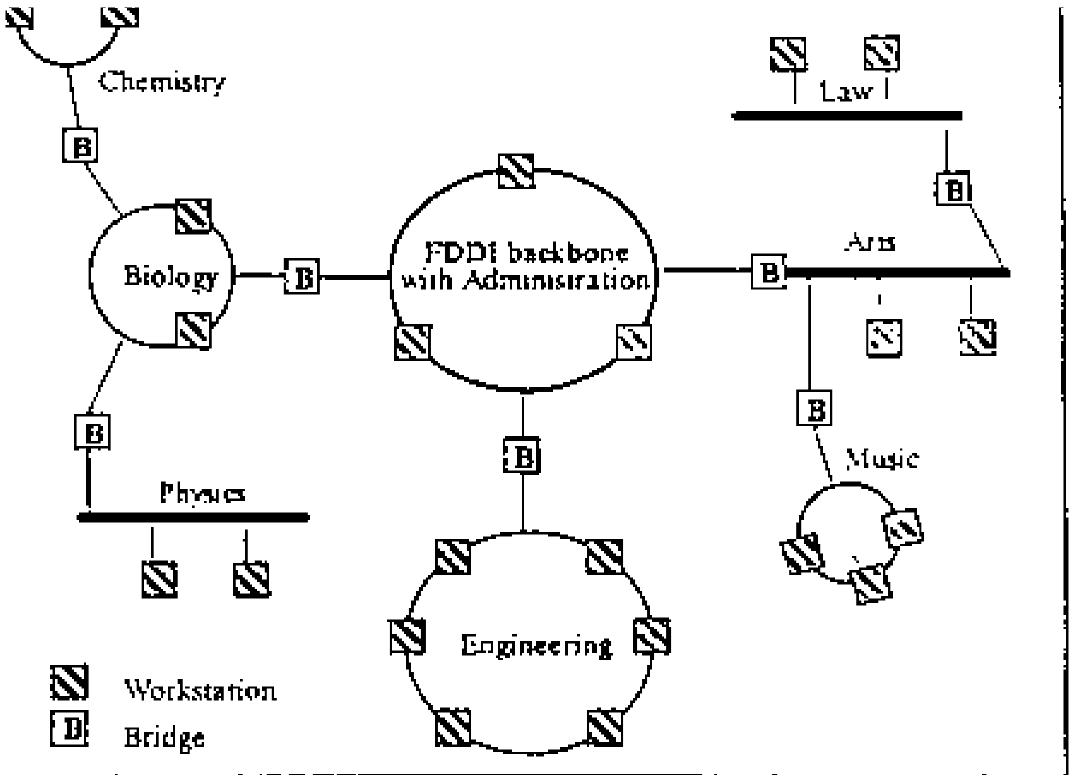


Figure 16: The Campus Network Under Study

The bridges in this model connect the various sub-networks to the backbone network. In this study, bridges perform basic operations, including buffering, filtering, and forwarding; they also perform protocol conversion of frames, if they bridge two dissimilar protocols. Bridge operation involves looking at all data frames and copying those with destination addresses on the opposite end of the bridge. A bridge will store a frame until it can be transmitted on the destination subnet according to the network protocol. Bridges may have finite buffer sizes, which causes packets to be rejected when the buffer is full.

An important aspect of this model is its use of the hierarchical approach to represent the campus network. At the top level, each subnet is represented by a single module. Subnet modules contain components that represent subnet nodes. Each node contains the specific traffic source and Media Access Control (MAC) model for the particular type of corresponding subnet. Two subnets, the Chemistry token ring subnet and the Arts Carrier Sense Multiple Access with Collision Detection (CSMA/CD) subnet, are shown in Figure 17 to illustrate a moderately complex hierarchy. Similar hierarchies are used for all subnets.

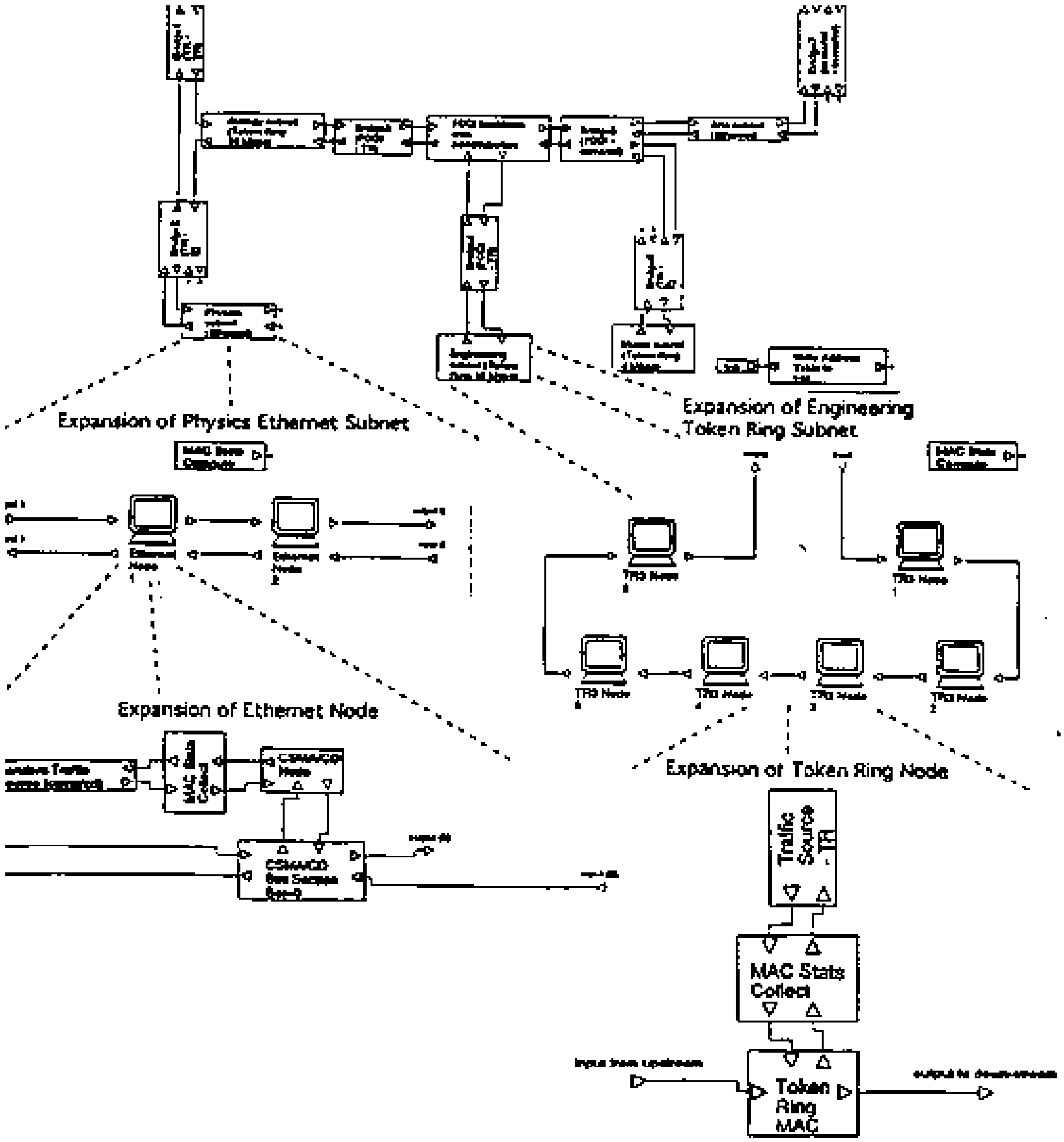


Figure 17: The Hierarchy of BONeS Models for the Campus Network Study

The focus of this example is on the evaluation of alternative network designs and their sensitivity analysis, that is, the performance of each design as a function of system parameters. In the specific scenario considered here, the concern is with the performance of the campus network as a function of user traffic. A simulation study over a set of eight traffic intensity values (see Table 4) was conducted by changing the mean interarrival time of requests at each node.

Table 4: Traffic Intensity Values for the Campus Network Simulation Study

Iteration	**Traffic Intensity (Mbps)**
1	0.400
2	0.468
3	0.548
4	0.641
5	0.749
6	0.877
7	1.025
8	1.199

The nonlinear performance as a function of traffic intensity gives a smaller separation between the "knee" of the curves over the range of loads of interest, marking the upper region of the system's operational range of interest.

Following are the values for the fixed parameters used in this study.

Packet Length	The length of the data packets was set to 6000 bits, without exception. All three MAC protocols add framing bits to determine the length of the frame they must transmit.
Mean Interarrival Time	This is an exported parameter from the Poisson pulse generator inside the traffic source in each workstation. It defines the average spacing between generated pulses in seconds. Since the data generation rate per node is specified by the traffic intensity parameter, the mean interarrival time then equals (packet length)/(traffic intensity).
CSMA/CD Propagation Delay	This parameter specifies the time it will take one bit to propagate between two adjacent CSMA/CD nodes. It was set to 1 microsecond.
FDDI Propagation Delay	This parameter specifies the time it will take one bit to propagate between two adjacent FDDI nodes. It was set to 10 microseconds.

Token Ring Propagation Delay	This parameter specifies the time it will take one bit to propagate on the Token Ring networks. It was set to 0.25 microsecond.
Maximum Queue Size	This is an exported parameter from the queue within each workstation. It specifies the maximum number of packets that can be stored in each workstation. It was set to 2000 packets.
Ring Latency (FDDI)	This parameter was set to 60.06 microseconds. It was calculated with the equation: total propagation time around the FDDI ring + (No. of nodes)*(one bit time for hardware processing delay).
Token Holding Time (workstation)	This parameter was set to 2.5 microseconds for 4 Mbps rings and 10 microseconds for 16 Mbps rings. It specifies the token holding time by a workstation in a Token Ring network.
T_Pr(0)	This parameter is the threshold time for asynchronous traffic with priority 0 in an FDDI network. It was set to 10 msec, which is also the Target Token Rotation Time for FDDI.

TTRT	This parameter is the Target Token Rotation Time for the FDDI network. The average rotation of the token on the FDDI backbone cannot exceed this parameter. The maximum time an FDDI node will wait before receiving a token is 2 * TTRT. TTRT was set to 10 msec.
TSTOP	This parameter specifies the length of simulated real time for each run. It was set to 10 seconds.

The sensitivity analysis comprising the study called for evaluating the throughput, delay, queue size, and bridge delay for the given design as a function of the traffic intensity at each node.

Figure 18 displays the throughput of each subnet plotted against the traffic intensity per node. The throughput of subnets with a smaller channel capacity (Chemistry and Music subnets) were the first to saturate as the traffic intensity increased beyond a critical point, around 0.64 Mbps.

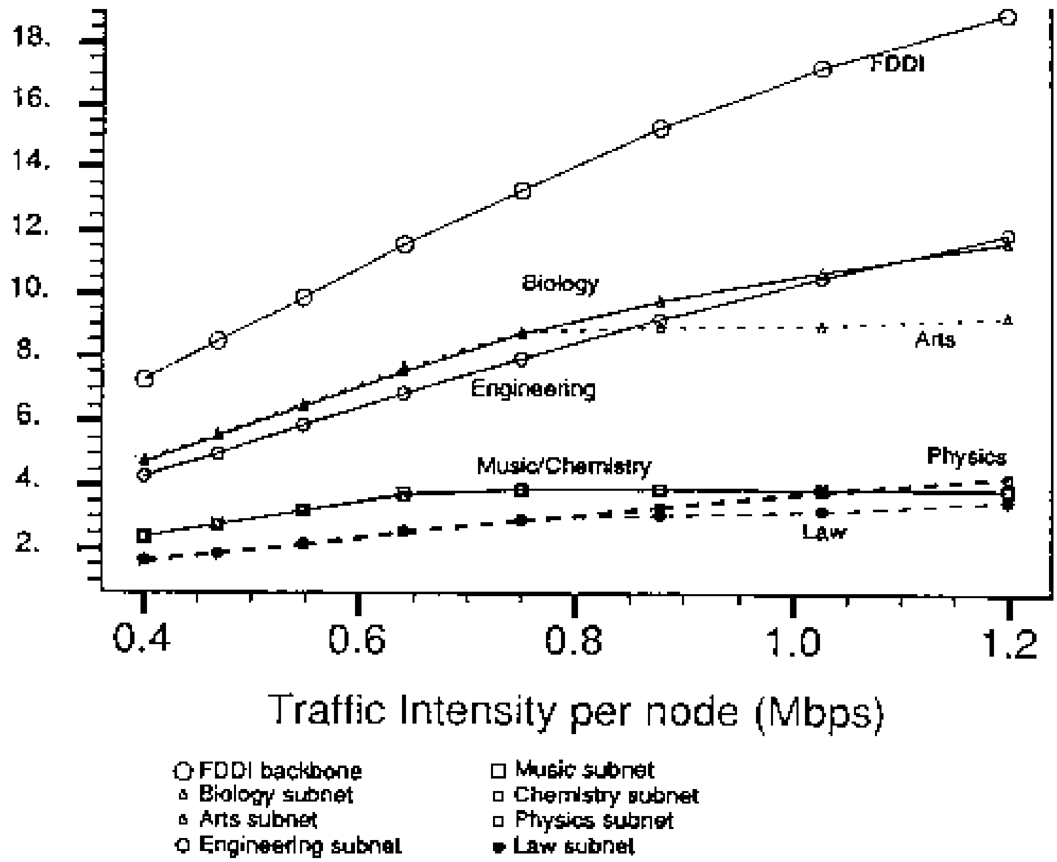


Figure 18: Throughput of Campus Network Model

Traffic entering the Biology and Chemistry subnets through the bridge connecting both subnets reached a maximum due to the saturation of the Chemistry subnet. Further increase in traffic intensity per node would thus prohibit any additional traffic flow from the Chemistry subnet to the Biology subnet. The slope of the throughput curve for the Biology subnet decreases concomitantly.

Since the Chemistry subnet transmits more traffic than the network can handle, packets are queued for transmission within the bridge. Figure 19 shows how the queue size grows as a function of load. As expected, at high loads the queue size grows steadily, until the buffer is filled to capacity; any further incoming packets are then rejected. A rough estimate of the maximum queueing delay of a packet encountering a full queue equals the average time to transmit 2000 packets on the 16 Mbps Biology subnet (i.e., 3.75 seconds). Figure 20 shows that the mean queueing delay for the bridge between Biology and Chemistry is close to 3.5 seconds, providing a sanity check.

The throughput results (Figure 18) also show that the Arts subnet saturates when the traffic intensity at each node reaches about 0.76 Mbps. When a number of subnets saturate simultaneously, the actual amount of traffic injected into the network becomes limited and, consequently, the throughput rate of increase of larger networks slows down at higher traffic intensity per node. This effect can be seen in the throughput curves for the FDDI backbone and Biology subnet.

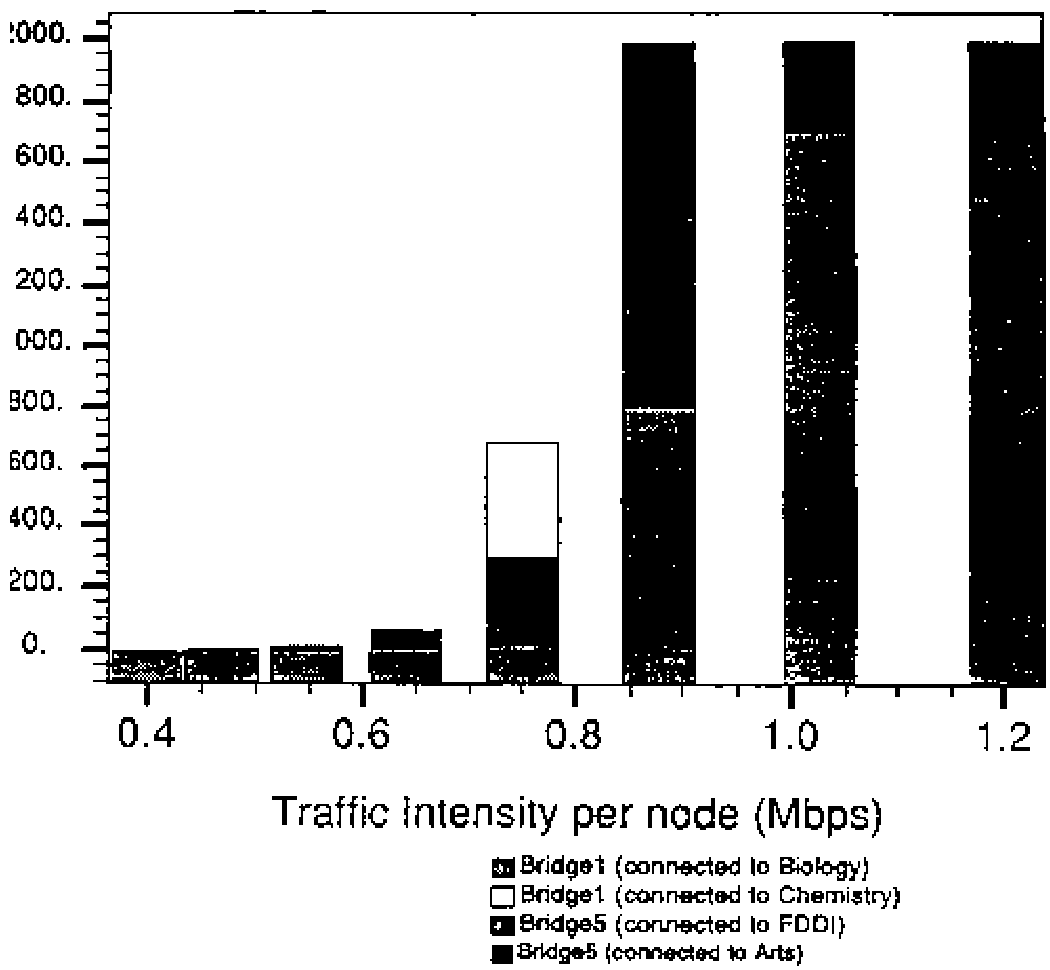


Figure 19: Queue Size in the Campus Network Model

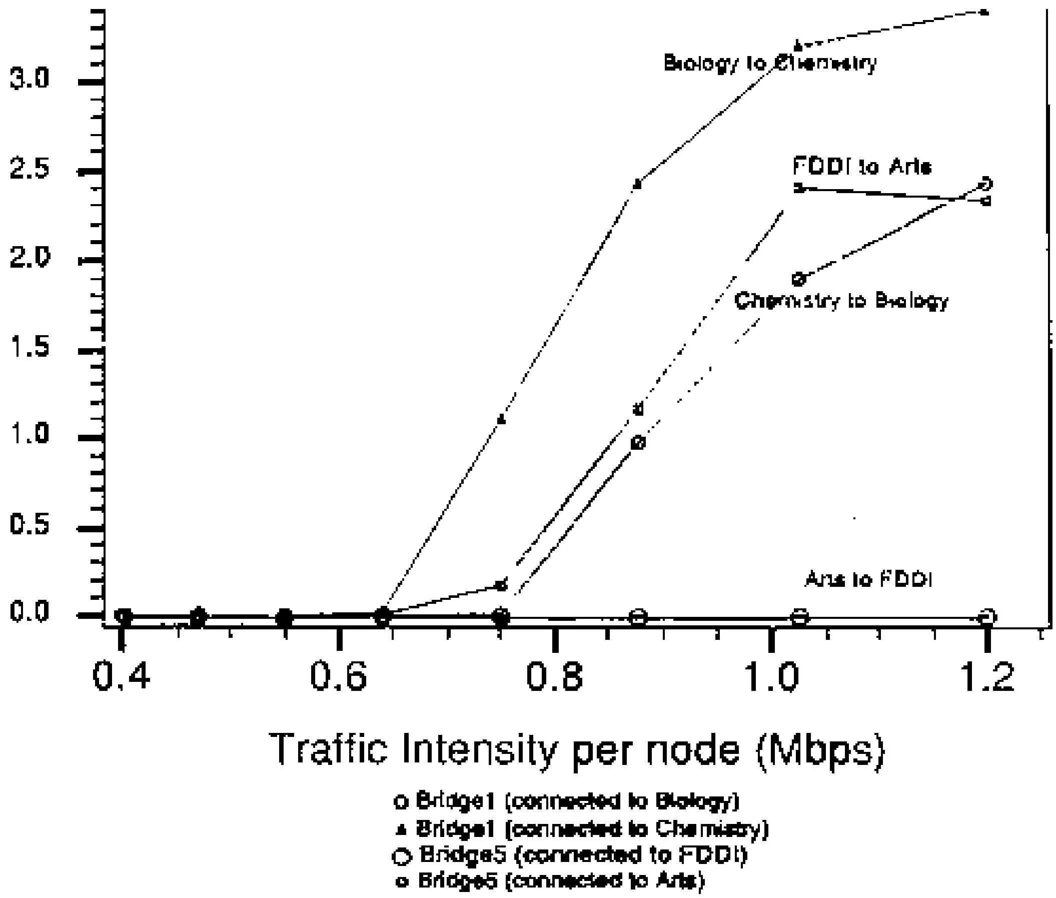


Figure 20: Bridge Queueing Delay in the Campus Network Model

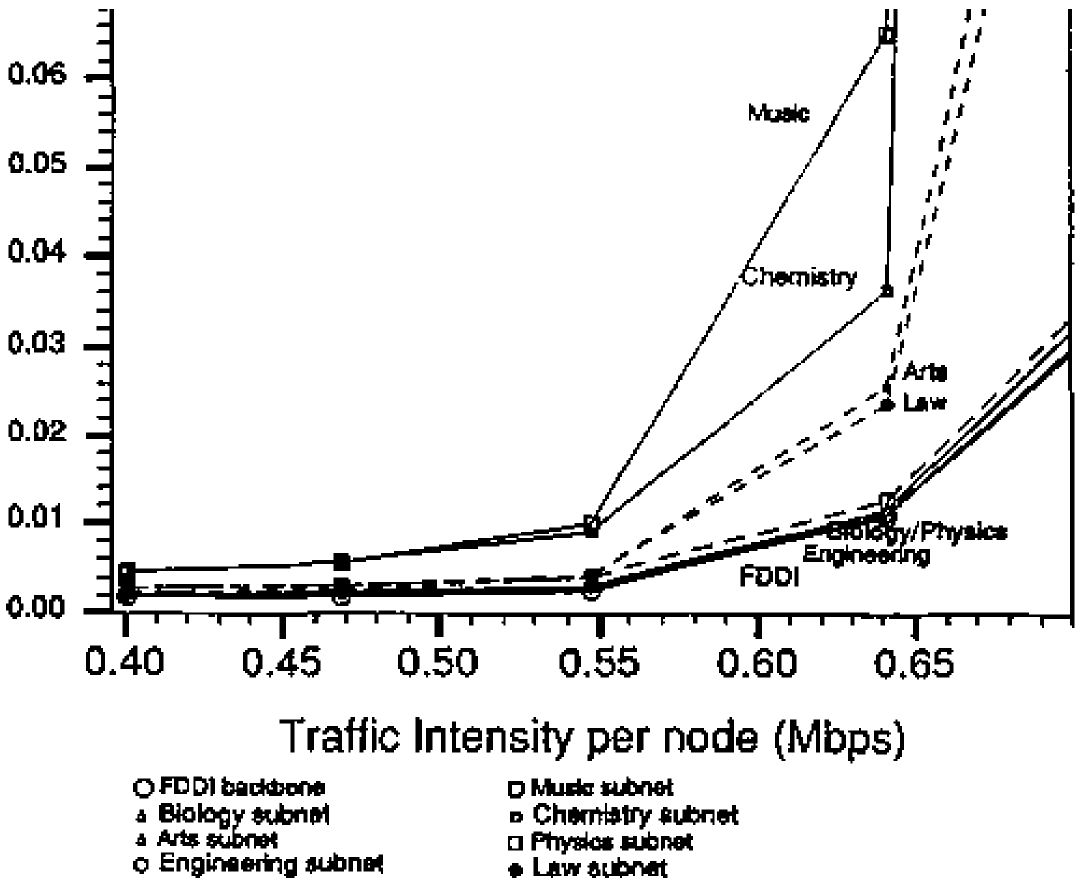


Figure 21: Subnet Queueing Delay in the Campus Network Model

Increasing queue sizes render the queueing delay the dominant factor in end-to-end transmission at high traffic intensity regimes. This effect is reflected in Figures 20 and 21 by the dramatic increase in delay statistics for both end-to-end subnet transmission delay and queueing delay through bridges.

The foregoing results provide an indication of the suitability of a given system design for satisfying user requirements. If user requirements are not met by a given design, then system topology and parameters can be easily changed, and the resulting configuration analyzed via a corresponding set of experiments.

7. SOME GUIDELINES TO SIMULATION MODELING AND ANALYSIS OF COMMUNICATIONS NETWORKS

It should be clear from the foregoing discussion that successful application of simulation technology to network design and management tasks is neither algorithmic nor amenable to mere mechanical application of tools. The purpose of this section is to present some guidelines to the successful use of simulation in the context of communications networks. General pitfalls and common errors involved in DES modeling are discussed in [12, 41, 42]. We stress, however, that no amount of theory and guidelines can supplant practical, hands-on experience.

Elements of a successful simulation study include:

- Know the customer

Clear objectives should be formulated at the outset concerning the goals of the simulation study, though some evolution of objectives is often inevitable in practice. Simulation should be used to address specific, well-defined questions and issues. The end consumer of the study, be it a network designer, technical network manager, or higher-level decision maker, should be aware of its goals and limitations; it is important to have a match of expectations between study providers and consumers. Performance metrics should be agreed upon and clearly defined. Simulation studies are costly, consuming resources such as manpower, equipment, and computer time. As in any economically driven enterprise, there should be a match between the goals of the simulation study and the allocated resources. Note that statistically-improved results can generally be obtained by allocating more resources to the study, especially longer runs or more detailed models. However, developing and implementing detailed models can be very time-consuming. A recurring error, common to many software projects, is to underestimate the time and effort required to carry out a simulation study. The consumer of the simulation project should be part of the modeling study throughout the lifetime of the project. Such consumers should be aware of the status of the project and be encouraged to provide constant feedback.

- Know the System/Network

A detailed understanding of the network under study is crucial. This knowledge is required to develop an appropriate system model. A common error is the inclusion of too much inconsequential detail in the model. This wastes precious manpower during model development, as well as computer time during production runs. Comprehension of network operation will aid in focusing the effort on the relevant components and detail. As suggested in [12], sensitivity studies as well as *network experts* can be employed to determine the appropriate level of modeling detail. In particular, it is critical to understand the random nature of demands on the network; thus, solid traffic modeling is essential to the success of the study. An arbitrary use of traffic models can easily lead to erroneous results, even as system internals are accurately modeled. Inappropriate models (e.g., modeling highly bursty traffic as a Poisson process) can result in performance predictions that miss the true performance by orders of magnitude. Knowledge of system operation will also facilitate the verification and validation process. Verification confirms that the simulation algorithm was implemented as intended. Validation confirms that the model used has adequate fidelity. Clearly, the more the analyst knows about the system, the easier it will be to verify and validate a model of that system.

- Know the Important Performance Metrics

Model detail, as well as the length of simulation runs, depend on the desired performance metrics. For example, rare-event simulations (e.g., estimating low packet loss probabilities) often require very long, or even prohibitively long, simulation runs, while estimating their average delay and throughput requires much shorter simulation runs. Since execution times may be quite long (e.g. hours, days, or more), it is important to collect as much information as possible during each run for post-processing analysis. Again, there must be a match between the objectives of study consumers and the simulation results. For example, a common specification is of the form "*the delay must be less than X ms 99% of the time*." This specification clearly indicates the nature of the performance metric expected from the simulation study. Simulation is often used to generate performance curves (e.g., delay vs. load), rather than single performance points. In this case, it is recommended that parametric studies be spelled out before the model is executed. Simulation studies are usually aimed at exploring a region of the response surface (network performance as a function of a number of variables). Understanding the critical segments of the response surface prior to the study enhances its likelihood of success. Application of the response surface methodology [90] and standard experimental design techniques [12] can be helpful.

Simulation is analogous to conducting an in-vitro experiment — the results being observations of random phenomena. The significance of variability in the results must be appreciated. A team member with statistical training would be very useful in interpreting the significance of simulation output. Making use of the sample variance or confidence intervals to characterize this variability is important in establishing the credibility of the results. Statistics from a single run (replication) of a model, or inadequately long runs, are often meaningless; similarly, the inappropriate use of the iid assumption on correlated data can lead to significantly erroneous results. State-of-the-art simulation environments provide an animation capability; that is, the flow of transactions (e.g., packets, messages, etc.) through the network can be visualized in real time. This is a valuable tool for model debugging and verification.

- Know How to Establish a Credible Model

A model is called credible if both its specification and predictions are accepted by study consumers as valid and used in the decision process [12]. As network management systems make increasing use of simulation, the decision process may deal with the operational aspects of the network as well as the traditional design planning activities. [12] presents a process to establish a valid and credible simulation model. An important step is to make sure that the model has a high *face value*, i.e., the results should *look* reasonable to knowledgeable people and pass various sanity checks. Favorable comparisons of the results to relevant theory, i.e., changing some of the model assumptions to satisfy available theory, will increase its credibility, as will supporting results from similar studies. Obviously, a favorable comparison of model predictions to empirical measurements from a real network will support model validity.

- Expect the Model to Evolve

A major advantage of simulation is that it can be used to generate performance predictions, even as network traffic patterns change and the underlying technology continues to evolve. It is important to realize from the beginning that the simulation model will not be static. Rather, it should be constructed with sufficient flexibility to track changes as necessary. Consumer satisfaction with simulation results often leads to requests for modifications to the model, e.g., a change in traffic, topology, or protocol. Thus, modularity and extendibility of models are important design features of any model.

- Know How to Apply Good Software Management Techniques

Simulation models are implemented as large computer programs. Thus, standard software development and change management techniques should be used to the extent possible. This process begins with the selection of an appropriate modeling environment or language. It includes proper verification of the transformation of the conceptual model into software. As discussed in [42], the design objectives, functional requirements, data structures, and progress must be constantly tracked. Such tracking can be aided by current software engineering tools. Design principles such as top-down design and structured programming should be applied to assist in the systematic development of the simulation model. The object-oriented programming paradigm is particularly suitable to communications network modeling. Incorporation of appropriate documentation is essential to the smooth evolution of the simulation model.

The above elements are needed to ensure the success of a communications network modeling project. They do not guarantee success, however, due to lack of experience and simulation's inherent limitations and pitfalls. Some of these are listed below.

- Simulation models can be expensive to construct and execute.

- Simulation does not give the insights obtainable from closed-form analytical results. Consequently, system optimization is difficult using simulation.

- Simulation results contain experimental error and can be misleading if not carefully analyzed and statistically interpreted.

As a last word of caution, the reader is reminded again that modeling is still considerably more art than science. Hands-on experience and willingness to learn from the inevitable mistakes are indispensable in bringing a simulation modeling project to a successful conclusion.

8. REFERENCES

1. Sauer, C.H. and K.M. Chandy, *Computer Systems Performance Modeling*. 1981, Englewood Cliffs, New Jersey: Prentice-Hall, Inc.

2. Schwartz, M., *Telecommunication Networks: Protocols, Modeling, and Analysis*. 1987, Reading, Massachusetts: Addison-Wesley.

3. Stuck, B.W. and E. Arthurs, *A Computer and Communications Network Performance Analysis Primer*. 1985, Englewood Cliffs, New Jersey: Prentice-Hall, Inc.

4. Reiser, M., *Performance Evaluation of Data Communication Systems.* Proceedings of the IEEE, 1982. **70**(2): p. 171-196.

5. Heidelberger, P. and S.S. Lavenberg, *Computer Performance Evaluation Methodology.* IEEE Transactions on Computers, 1984. **C-33**(12): p. 1195-1220.

6. Kleinrock, L., *Queuing Systems*. Vol. I and II. 1976, New York, New York: John Wiley & Sons, Inc.

7. Hayes, J.F., *Modeling and Analysis of Computer Communications Networks*. 1984, New York, New York: Plenum Press.

8. Ahuja, V., *Design and Analysis of Computer Communication Networks*. 1982, New York: McGraw-Hill, Inc.

9. Bertsekas, D.P. and R.G. Gallager, *Data Networks*. 1987, Englewood Cliffs, New Jersey: Prentice-Hall.

10. Gross, D. and C.M. Harris, *Fundamentals of Queuing Theory*. 2nd ed. 1985, New York: John Wiley & Sons, Inc.

11. Daigle, J.N., *Queuing Theory for Telecommunications*. 1992, Reading, Massachusetts: Addison-Wesley Publishing Company.

12. Law, A.M. and W.D. Kelton, *Simulation Modeling and Analysis*. 2nd ed. 1991, New York, New York: McGraw-Hill, Inc.

13. Cassandras, C.G., *Discrete Event Systems: Modeling and Performance*

Analysis. 1993, Homewood, Illinois: Irwin.

14. Hui, J.Y.N., *Switching and Traffic Theory for Integrated Broadband Networks*. 1990, Boston: Kluwer Academic Publishers.

15. Francis, S., V. Frost, and D. Soldan. *Measured Performance for Multiple Large File Transfers*. in *14th Conference on Local Computer Networks*. 1989. Minneapolis, Minnesota: IEEE.

16. Marshall, W.T. and S.P. Morgan, *Statistics of Mixed Data Traffic on a Local Area Network.* Computer Networks and ISDN Systems, 1985. **10**(3): p. 185-194.

17. Gusella, R., *A Measurement Study of Diskless Workstation Traffic on an Ethernet.* IEEE Transactions on Communications, 1990. **38**(9): p. 1557-1568.

18. Jain, R. and S.A. Routhier, *Packet Trains — Measurements and a New Model for Computer Network Traffic.* IEEE Journal on Selected Areas in Communications, 1986. **SAC-4**(6): p. 986-995.

19. Fishman, G.S., *Principles of Discrete Event Simulation*. 1978, New York, New

York: John Wiley & Sons, Inc.

20. Kurose, J.F. and C. Shen. *GENESIS: A Graphical Environment for the Modeling and Performance Analysis of Protocols in Multiple Access Networks*. in *ICC'86*. 1986. Toronto, Ontario, Canada: IEEE.

21. Schriber, T.J., *Simulation Using GPSS*. 1974, John Wiley & Sons, Inc.

22. Pritsker, A.A.B., *Introduction to Simulation and SLAM II*. 1984, New York, New York: John Wiley & Sons, Inc.

23. Law, A.M. and C.S. Larmey, *An Introduction to Simulation Using SIMSCRIPT II.5*. 1984, Los Angeles, California: C.A.C.I.

24. *MODSIM Reference Manual*. 1993, La Jolla, California: CACI Products Co.

25. Sauer, C.H., M. Reiser, and E.A. MacNair. *RESQ — A Package for Solution of Generalized Queuing Networks*. in *National Computer Conference*. 1977.

26. Sauer, C.H. and E.A. MacNair, *Simulation of Computer Communication*

Systems. 1983, Englewood Cliffs, New Jersey: Prentice-Hall, Inc.

27. Sauer, C.H., E.A. MacNair, and J.F. Kurose, *Queuing Network Simulation of Computer Communications.* IEEE Journal on Selected Areas in Communications, 1984. **SAC-2**(1): p. 203-220.

28. *PAWS/A', User Guide*. 1983, Austin, Texas: Information Research Associates.

29. Melamed, B. and R.J.T. Morris, *Visual Simulation: The Performance Analysis Workstation.* IEEE Computer, 1985. **18**: p. 87-94.

30. *SES/Workbench™ Introductory Overview*. 1989, Austin, TX: Scientific and Engineering Software, Inc.

31. Jerome, A.C., S.P. Baraniuk, and M.A. Cohen. *Communication Network and Protocol Design Automation*. in *IEEE Military Communications Conference*. 1987. Washington, D.C.: IEEE.

32. Marsan, M.A., *et al.*, *TOPNET: A Tool for the Visual Simulations of Communications Networks.* IEEE Journal on Selected Areas in

Communications, 1990. **8**(9): p. 1735-1747.

33. Dupuy, A. and e. al., *“NEST” A Network Simulation and Prototyping Testbed.* Communications of the ACM, 1990. **33**(10): p. 63-74.

34. LaRue, W.W., *et al*. *A Block Oriented Paradigm for Modeling Communications Networks*. in *IEEE Military Communications Conference*. 1990. Monterey, California: IEEE.

35. Van Norman, H.J., *LAN/WAN Optimization Techniques*. 1992, Boston, Massachusetts: Artech House.

36. Cope, P., *Building a Better Network,* in *Network World*1993, p. 34-36, 40.

37. Law, A.M. and M.G. McCommas, *Simulation Software for Communications Networks.* IEEE Communications Magazine, 1994. **32**(3).

38. Hess, J.L., *PlanNet Models Network Traffic,* in *Corporate Computing*1992, p. 45-46.

39. Cinlar, E., *Introduction to Stochastic Processes*. 1975, Englewood Cliffs, New Jersey: Prentice-Hall.

40. Crane, M.A. and A.J. Lemoine, *An Introduction to the Regenerative Method for Simulation Analysis*. 1977, Berlin: Springer-Verlag.

41. Pooch, U.W. and J.A. Wall, *Discrete Event Simulation: A Practical Approach*. 1993, Boca Raton, Florida: CRC Press.

42. Jain, R., *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. 1991, New York, New York: John Wiley & Sons, Inc.

43. Leland, W.E., *et al.*, *On the Self-Similar Nature of Ethernet Traffic*. 1993, Morristown, New Jersey: Bellcore.

44. Franken, P.D., U.A. Koenig, and V. Schmidt, *Queues and Point Processes*. 1981, Berlin: Akademie-Verlag.

45. Beran, J., *et al.*, *Variable-Bit-Rate Video Traffic and Long-Range Dependence.* IEEE Transactions on Communications, 1992. .

46. Fendick, K.W. and W. Whitt, *Measurements and Approximations to Describe the Offered Traffic and Predict the Average Workload in a Single-Server Queue.* Proceedings of the IEEE, 1989. **77**(1): p. 171-194.

47. Eckberg, A.E. *Generalized Peakedness of Teletraffic Processes*. in *Tenth International Teletraffic Conference*. 1983. Montreal, Canada:

48. Gusella, R., *Characterizing the Variability of Arrival Processes with Indexes of Dispersion.* IEEE Journal on Selected Areas in Communications, 1991. **9**(2): p. 203-211.

49. Livny, M., B. Melamed, and A.K. Tsiolis, *The Impact of Autocorrelation on Queuing Systems.* Management Science, 1993. **39**(3): p. 322-339.

50. Larson, H.J. and B.O. Shubert, *Probabilistic Models in Engineering Sciences*. Vol. I and II. 1979, New York, New York: John Wiley & Sons, Inc.

51. Lucantoni, D.M., K.S. Meier-Hellstern, and M.F. Neuts, *A Single-Server Queue with Server Vacations and a Class of Non-Renewal Arrival Processes. Adv. Appl. Prob.*, 1990. **22**: p. 676-705.

52. Heffes, H. and D.M. Lucantoni, *A Markov Modulated Characterization of Packetized Voice and Data Traffic and Related Statistical Multiplexer Performance.* IEEE Journal on Selected Areas in Communications, 1986. **SAC-4**(6): p. 856-868.

53. Takine, T., B. Sengupta, and T. Hasegawa, *An Analysis of a Discrete-Time Queue for Broadband ISDN with Priorities Among Traffic Classes*. IEEE Transactions on Communications, 1993. **to appear**.

54. Anick, D., D. Mitra, and M.M. Sondhi, *Stochastic Theory of a Data-Handling System with Multiple Sources.* The Bell System Technical Journal, 1982. **61**(8): p. 1871-1894.

55. Kobayashi, H. and Q. Ren, *A Mathematical Theory for Transient Analysis of Communications Networks.* IEICE Transactions on Communications,, 1992. **E75-B**(12): p. 1266-1276.

56. Sen, P., *et al.*, *Models for Packet Switching of Variable-Bit-Rate Video Sources.* IEEE Journal on Selected Areas in Communications, 1989. **7**(5): p. 865-869.

57. Box, G.E.P. and G.M. Jenkins, *Time Series Analysis: Forecasting and Control*. 1976, Englewood Cliffs, New Jersey: Prentice-Hall.

58. Heyman, D.P., A. Tabatabai, and T.V. Lakshman, *Statistical analysis and simulation study of video teleconference traffic in ATM networks.* IEEE Transactions on Circuits and Systems for Video Technology, 1992. **2**(1): p. 49-59.

59. Ramamurthy, G. and B. Sengupta. *Modeling and Analysis of a Variable Bit Rate Video Multiplexer*. in *INFOCOM'92*. 1992. Florence, Italy:

60. Jagerman, D.L. and B. Melamed, *The Transition and Autocorrelation Structure of TES Processes; Part II: Special Cases.* Stochastic Models, 1992. **8**(3): p. 499-527.

61. Jagerman, D.L. and B. Melamed, *The Transition and Autocorrelation Structure of TES Processes; Part I: General Theory.* Stochastic Models, 1992. **8**(2): p. 193-219.

62. Melamed, B. and B. Sengupta, *TES Modeling of Video Traffic.* IEICE Transactions on Communications, 1992. **E75-B**(12): p. 1292-1300.

63. Devroye, L., *Non-Uniform Random Variate Generation*. 1986, New York: Springer-Verlag.

64. Geist, D. and B. Melamed. *TEStool: An Environment for Visual Interactive Modeling of Autocorrelated Traffic*. in *International Conference on Communications*. 1992. Chicago, Illinois:

65. Lee, D.S., *et al.*, *TES Modeling for Analysis of a Video Multiplexer.* Performance Evaluation, 1992. **16**: p. 21-34.

66. Reibman, A.R., *DCT-Based Embedded Coding for Packet Video.* Signal Processing: Image Communication, 1991. **3**: p. 231-237.

67. Le Gall, D., *MPEG: A Video Compression Standard for Multimedia Applications.* Communications of the ACM, 1991. **34**(4): p. 46-58.

68. Li, S.-Q. and C.-L. Hwang. *Queue Response to Input Correlation Functions: Discrete Spectral Analysis*. in *INFOCOM'92*. 1992. IEEE.

69. Li, S.-Q. and C.-L. Hwang. *Queue Response to Input Correlation Functions: Continuous Spectral Analysis*. in *7th Computer Communication Workshop*. 1992. Hilton Head Island, South Carolina:

70. Sheng, H.-D. and S.-Q. Li. *Second Order Effect of Binary Sources on Characteristics of Queue and Loss Rate*. in *INFOCOM'93*. 1993. IEEE.

71. Sheng, H.-D. and S.-Q. Li. *Spectral Analysis of Packet Loss Rate at Statistic Multiplexer for Multimedia Services*. in *GLOBECOM'92*. 1992. IEEE.

72. Ye, J. and S.-Q. Li, *Folding Algorithm: A Computational Method for Finite QBD Processes with Level-Dependent Transitions.* IEEE Transactions on Communications, 1994. **42**(2).

73. Li, S.-Q. and S. Chong. *Fundamental Limits of Input Rate Control in High-Speed Network*. in *INFOCOM'93*. 1993. IEEE.

74. Li, S.-Q., *et al*. *Link Capacity Allocation and Network Control by Filtered Input Rate in High Speed Networks*. in *GLOBECOM'93*. 1993. IEEE.

75. Mandelbrot, B.B., *The Fractal Geometry of Nature*. 1983, New York, New

York: Freeman.

76. Mandelbrot, B.B. and J.W. Van Ness, *Fractional Brownian Motions, Fractional Noises and Applications.* SIAM Review, 1968. **10**(4): p. 422-437.

77. Grange, C.W.J. and R. Joyeux, *An Introduction to Long-Memory Time Series Models and Fractional Differencing.* Time Series Anal., 1980. **1**: p. 15-29.

78. Cox, D.R., *Long-Range Dependence: A Review,* in *Statistics: An Appraisal,* H.A. David and H.T. David, Editor^Editors. 1984, The Iowa State University Press: Ames, Iowa. p. 55-74.

79. Hurst, H.E., *Long-Term Storage Capacity of Reservoirs.* Transactions of the American Society of Civil Engineers, 1951. **116**: p. 770-799.

80. Balaban, P. and K.S. Shanmugan, *Computer-Aided Modeling Analysis and Design of Communications systems: An Introduction and Issue Overview.* IEEE Journal on Selected Areas in Communications, 1984. **SAC-2**(1).

81. Jackman, J. and D.J. Medevias, *A Graphical Methodology for Simulating Communications Networks.* IEEE Transactions on Communications, 1988.

36(4).

82. Kurose, J.F. and H.T. Mouftah, *Computer-aided Modeling, Analysis and Design of Communication Systems.* IEEE Journal on Selected Areas in Communications, 1988. (1).

83. Green, P.E., Jr., *Computer Network Architectures and Protocols*. 1982, New York, New York: Plenum Press.

84. Garg, K., *An Approach to Performance Specification of Communication Protocols Using Timed Petri Nets.* IEEE Transactions on Software Engineering, 1985. **SE-11**(10): p. 1216-1225.

85. Sanders, W.H. and J.F. Meyer, *Reduced Base Model Construction Methods for Stochastic Activity Networks.* IEEE Journal on Selected Areas in Communications, 1991. **9**(1): p. 25-36.

86. Movaghar, A. and J.F. Meyer. *Performability Modeling with Stochastic Activity Networks*. in *Real-Time Systems Symposium*. 1984. Austin, Texas:

87. Meyer, J.F., A. Movaghar, and W.H. Sanders. *Stochastic Activity Networks:*

Structure, Behavior, and Application. in *International Workshop on Timed Petri Nets*. 1985. Torino, Italy: IEEE.

88. Billington, J., G.R. Wheeler, and M.C. Wilbur-Ham, *PROTEAN: A High-level Petri Net Tool for the Specification and Verification of Communication Protocols.* IEEE Transactions on Software Engineering, 1988. **14**(3): p. 301-316.

89. Shanmugan, K.S., W.W. LaRue, and V.S. Frost, *A Block-Oriented Network Simulator (BONeS).* Journal of Simulation, 1992. **58**(2).

90. Myers, R.H., *Response Surface Methodology*. 1971, Boston, Massachusetts: Allyn & Bacon.

ACKNOWLEDGMENTS

The authors thank San-Qi Li for his invaluable contribution to the section on the frequency domain approach.

ABOUT THE AUTHORS

Victor S. Frost is a Professor of Electrical Engineering and Computer Science and Director of the Telecommunications and Information Sciences Laboratory at the University of Kansas. He joined the faculty of the University of Kansas in 1982. In 1981 he was a Visiting Scientist at the German Aerospace Research Establishment (DFVLR), Oberpfaffenhofen, West Germany. In 1985 he was a resident visiting member of the technical staff at AT&T Bell Laboratories, Denver, Colorado. He was chairman of the Kansas City Chapter of the Communications Society in 1992. His current research interest is in the areas of integrated communication networks, high-speed telecommunications networks, communications system analysis, and simulation. Dr. Frost received a Presidential Young Investigator Award from the National Science Foundation in 1984, an Air Force Summer Faculty Fellowship, a Ralph R. Teetor Educational Award from the Society of Automotive Engineers, and the Miller Professional Development Awards for Engineering Research and Service in 1986 and 1991, respectively. He is a member of Eta Kappa Nu and Tau Beta Pi, and he is a Senior member of the IEEE.

Benjamin Melamed is Head of the Performance Analysis Department at the C&C Research Laboratories, NEC USA, Inc., at Princeton, New Jersey, where he has been employed since 1989. His research interests include system modeling and analysis, simulation, stochastic processes, and visual modeling environments. Melamed received a B.Sc. degree in Mathematics and Statistics from Tel Aviv University in 1972 and M.S. and Ph.D. degrees in Computer Science from the University of Michigan in 1973 and 1976, respectively. From 1977 to 1981 he taught at the department of Industrial Engineering and Management Science at Northwestern University. He joined the Performance

Analysis Department at Bell Laboratories in 1981 and became an AT&T Bell Laboratories Fellow in 1988. Melamed was elected an IEEE Fellow in 1994.