Timestep Stochastic Simulation of Computer Networks using Diffusion Approximation

Andrzej Kochut *
IBM T.J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598, USA
akochut@us.ibm.com

A. Udaya Shankar
Department of Computer Science
University of Maryland
College Park, MD 20742
shankar@cs.umd.edu

Abstract

Timestep stochastic simulation (TSS) is a novel method for generating sample paths of computer networks, with low computation cost independent of packet rates. It has accuracy adequate to evaluate general network and flow configurations, including arbitrary flow start times and durations, drop-tail queuing (i.e., does not require RED), and arbitrary state-dependent control mechanisms for congestion control and routing. TSS generates the evolution of the system state $S(t)$ on a sample path in time steps of size δ. At each step, $S(t+\delta)$ is randomly chosen according to $S(t)$ and the probability distribution $Pr[S(t+\delta)|S(t)]$ obtained using the diffusion approximation. Because packet transmission and reception events are replaced by time steps, TSS generates sample paths at a fraction of the cost of packet-level simulation. Because TSS generates sample paths, control feedback can be based on sample path metrics, rather than ensemble metrics, thereby accurately capturing the effects of state-dependent control mechanisms.

1 Introduction

Performance evaluation is crucial to advancing the state of computer networking. Yet existing evaluation techniques are not adequate for handling the large heterogeneous architectures and state-dependent control schemes that are intrinsic to modern computer networking. A state-dependent control scheme is one where control (e.g., source rate or window size, next hop, buffering threshold) is exercised based on the observed state of the network (typically averaged over some recent interval).

Packet-level simulation (e.g. [32, 30]), currently the most popular technique, is accurate but does not scale with link bandwidth and network size. Purely analytical techniques (e.g. [13, 14]) do not capture the effects of state-dependent control or realistic traffic mix with reasonable accuracy.

This current state of affairs has motivated techniques such as fluid approximation models (e.g. [25, 20, 3, 2, 1, 21]), henceforth referred to as "fluid" methods, where the arrival and service times are defined by time-varying means, and a timestep computation yields an evolution of the network state.

TSS can be viewed as a generalization of fluid approximation methods, based on the first two moments rather than only the first moment. In particular, arrival and service times in TSS are defined by time-varying rate *and* variance, and a TSS computation yields *an* evolution of the network state.

As in stochastic packet-level simulation, repeated executions of TSS on the same network (including the same arrival and service parameters) yield *different* evolutions, each possible evolution corresponding to a particular choice of random values (e.g., arrival or service times). The time evolution of the first two moments of desired metrics can be obtained by computing multiple evolutions and averaging (as in stochastic packet-level simulation).

It is worthwhile to emphasize the distinction between TSS and fluid approximation methods. Both TSS and fluid methods compute an evolution. Fluid methods use deterministic description of both inter-arrival and service processes. Thus repeated executions of a fluid method on the same network (including the same arrival and service parameters) yield the *same* evolution. Whereas in the case of TSS (and stochastic packet-level simulation), repeated executions on the same network yield *different* evolutions.

Ultimately, we are interested not in individual evolutions but in the evolution of instantaneous ensemble metrics, e.g., the evolution of mean queue size. In TSS and in packet-level simulation, this is obtained by computing multiple evolutions and averaging over them. Whereas a

*Work done when author was a graduate student at the University of Maryland, College Park.

fluid method computes only one evolution. So the question arises as to when this one evolution adequately represents all the different possible evolutions. Clearly, this would hold if the different possible evolutions are all "close" to each other. While this may be true of certain types of computer networks, it is not true of most computer networks, especially when the link scheduling is drop-tail as opposed to a "per-flow" discipline like RED. The nonlinear control mechanisms present in networks invariably cause small differences to lead to vastly different evolutions over time; for example, a small delay in a packet arrival may trigger a timeout, after which the evolution would be very different than if the timeout had not occurred. Section 2 illustrates the diversity of queue size evolutions and its impact on fluid approximations using a simple example network.

Timestep Stochastic Simulation (TSS)

Our method computes the evolution of system state $S(t)$ on a sample path at time instants t_0, t_1, t_2, $\cdots$, where $t_{i+1} - t_i = \delta$ for an appropriately small δ (smaller than the time scale of the end-to-end control) assuming a starting state $S(t_0)$. Note that $S(t)$ contains the values of all important system variables at time t, for example queue sizes of communication links, window sizes of TCP connections, state of TCP connections (i.e., slow start or congestion avoidance), etc. $S(t_{i+1})$ can be computed based on $S(t)$ and models of congestion control mechanisms.

To illustrate the idea consider one of the variables in $S(t)$, the number of customers $N(t)$ of a queue. We do the following for $i = 0, 1, \cdots$:

(1) Analytically obtain $Pr[N(t_{i+1})|N(t_i)]$, the probability distribution of $N(t_{i+1})$ given $N(t_i)$.

(2) Then *randomly* choose a value for $N(t_{i+1})$ based on $Pr[N(t_{i+1})|N(t_i)]$.

Repeating these two steps for successive intervals generates a sample path of the queue size $N(t)$. Figure 1 depicts this procedure. Assume, that the initial queue size at time t_0 equals $N(t_0)$ (marked by a black dot). Because of the randomness in arrival and service processes the time evolution of $N(t)$ in the interval $[t_0, t_0 + \delta]$ can follow a multiplicity of paths. Two of these paths originating from $(t_0, N(t_0))$ are shown. Each upward transition corresponds to the arrival of the packet to the queue and each downward transition corresponds to the departure of the packet. The queue size $N(t_0 + \delta)$ is chosen based on the probability density of queue size at $t_0 + \delta$ conditioned on $N(t_0)$. The probability density is shown as a solid line. The procedure is repeated for step $[t_0 + \delta, t_0 + 2\delta]$, as shown in the figure.

The difficulty, of course, is in obtaining an analytical expression for $Pr[N(t_{i+1})|N(t_i)]$. The arrival and service distributions in the interval $[t_i, t_{i+1}]$ are available from

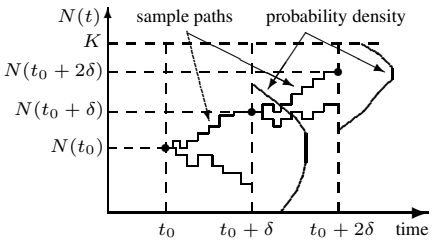


Figure 1: Computation procedure. At each step, the probability distribution of $N(t + \delta)$ given the current value of $N(t)$ is obtained and a new value of $N(t + \delta)$ is randomly chosen based on this distribution.

$S(t_i)$ and models of control schemes (such as TCP). Even so, exact results for $Pr[N(t_{i+1})|N(t_i)]$ are not known for most queuing systems of interest. However, the diffusion approximation, first proposed by Kolmogorov [18] and later extended by Feller [7], provides a very accurate approximation of this distribution.

The diffusion approximation has two critical advantages. First, packet arrival and service distributions can be time-varying stochastic processes characterized by the first two instantaneous moments, i.e., mean rate at time t and variance at time t. This allows realistic modeling of traffic and service times. In particular, the variation of a source at time t represents how much jitter the source exhibits with respect to its mean rate at time t. Second, because the diffusion approximation is based on the law of large numbers, its accuracy increases with increasing packet rates (and its computational cost is not affected).

TSS is a general-purpose network simulator whose computational cost does not increase with increasing bandwidth. It can be used as a general substitute for packet-level simulation of high speed networks, for example, not only to examine special problems like the behavior of many TCP flows sharing a bottleneck, but also to evaluate P2P networks, load balancing, interaction between dynamic routing and flows, and so on. This paper focuses on handling an arbitrary network topology with drop-tail queuing. However, TSS can be easily extended to the statistically smoother AQM disciplines. TSS can handle both the TCP and UDP flows with arbitrary start times and durations (i.e., both TCP elephants and mice [21]).

2 Diversity of Sample Paths and its Impact on Deterministic Fluid Approximations

We now present an example that illustrates the diversity of evolutions found in even simple networks, show how well TSS handles this diversity, and discuss what are the implications for fluid modeling.

Figure 2 shows a two-link tandem network with TCP flow from node N_0 to node N_2. Both links have equal ser-

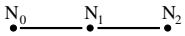


Figure 2: Two-link network.

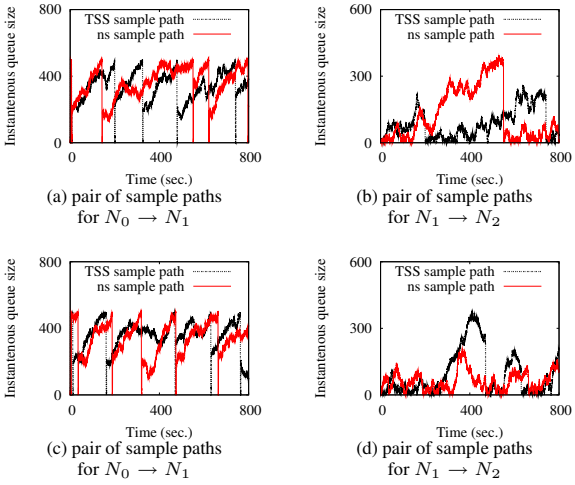


Figure 3: Evolutions of queue size generated by packet-level simulation and by TSS for tandem network in Figure 2.

vice rate of 1000 packets/sec, squared coefficient of variation of service time of 0.05, and buffer capacity of 500 packets. Figures 3a and 3c show two pairs of evolutions as generated by packet-level simulation and TSS for link $N_0 \rightarrow N_1$. Similar pairs for link $N_1 \rightarrow N_2$ are shown in Figures 3b and 3d. Figures 4a and 4c present evolutions of mean queue size and its standard deviation for link $N_0 \rightarrow N_1$, respectively. Figures 4b and 4d show corresponding evolutions for link $N_1 \rightarrow N_2$.

There are several points to note. First, the individual evolutions have great diversity, especially those of link $N_1 \rightarrow N_2$ (due to the first queue being nearly always non-empty). There is no "representative" evolution. Second, TSS is accurate in computing individual evolutions as well as the evolutions of ensemble metrics, for both links. TSS is 10 times faster than ns [32] for the presented configuration. The difference increases to nearly 1000 times in case of two 1Gbps links. Third, the instantaneous mean queue size evolution of a link is quite different from any of its individual evolutions; the cyclical nature of the latter is not present in the former. A fluid method would not have the random time shifts that make the mean queue size evolution smooth (as presented on Figure 4a). Instead, it would yield a cyclical "saw-tooth" behavior that does not decay with time and that corresponds to only one of the many possible evolutions; the evolution would not be a representative evolution, say in the example depicted on Figure 3a. This problem stems from the simplification (used in all fluid approximations) that the behavior of traffic sources is driven by the

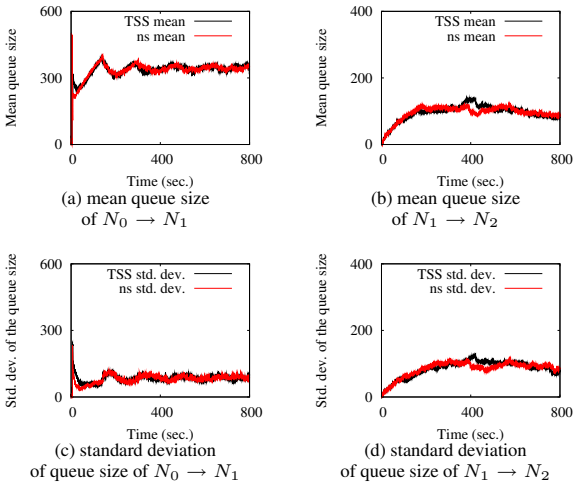


Figure 4: Mean queue size and its standard deviation for link $N_0 \rightarrow N_1$ and for link $N_1 \rightarrow N_2$ of network in Figure 2 as obtained by packet-level simulation and by TSS.

mean system evolution and not the sample system evolution. The deficiency of fluid models is even more visible in case of the second queue in tandem. Fluid approximation predicts that this queue is empty all the time, because arrival rate never exceeds the service rate of the link. In reality however, randomness in inter-arrival and service processes cause the queue to impact the flows' round-trip time significantly. TSS predicts this behavior very well as shown on Figures 4b and 4d. Fourth, TSS yields the evolution not only of mean values but also of second moments and even distributions of metrics of interest.

3 TSS for One Queue

We model a communication link by a single-server queue in the usual way, with service rate equal to the link bandwidth (in packets/second) and maximum queue size equal to the link buffer size (in packets). The state of the link at time t is defined by the number of packets in the queue at time t, denoted by $N(t)$.

We compute a time-stepped evolution of the stochastic process $N(t)$ given a starting state $N(t_0)$ as illustrated in Figure 1. Divide the time axis into intervals defined by time instants $t_0, t_1, \cdots$ where $t_{i+1} - t_i = \delta$ and δ is smaller than the control time scale Δ. Then do the following iteratively for $i = 0, 1, \cdots$: compute $Pr[N(t_{i+1})|N(t_i)]$ and choose a random $N(t_{i+1})$ with this distribution. In order to compute $Pr[N(t_{i+1})|N(t_i)]$ we use diffusion approximation [18, 7] with holding barriers [10] (with Coxian distributions at the barriers [4]).

We next present the diffusion approximation equations for an unbounded queue, i.e., with only a lower barrier

$N(t) = 0$. Similar equations may be obtained for the two-barrier case ($N(t) = 0$ and $N(t) = K$), but the details are omitted here for brevity.

Let us introduce the following notation:

- $f(x, t|x_0)$: conditional probability density of $N(t) = x$ given $N(0) = x_0$.
- λ and c_A: mean rate and squared coefficient of variation of the arrival process.
- μ and c_S: mean rate and squared coefficient of variation of the service process.
- λ_i and a_i: exit rate and probability of entering next stage, for ith stage of Coxian holding time distribution.
- $P_i(t)$: probability of being in ith stage of Coxian distribution at time t.

The time evolution of $f(x, t|x_0)$ is the solution to the diffusion equation, where $f(x, t|x_0)$ is replaced with f for brevity:

$$\frac{\partial f}{\partial t} = \frac{\alpha}{2}\frac{\partial^2 f}{\partial x^2} - \beta\frac{\partial f}{\partial x} + \sum_{i=1}^{n} P_i(t)\delta^*(x-1)\lambda_i(1-a_i) \quad (1)$$

where

$$\begin{aligned} \beta &= \lambda - \mu \\ \alpha &= \lambda c_A + \mu c_S. \end{aligned} \quad (2)$$

and initial and boundary conditions are as follows ($\delta^*(x)$ is the Dirac delta function):

$$\begin{aligned} f(0, t|x_0) &= 0 \qquad t \geq 0 \\ f(x, 0|x_0) &= \delta^*(x - x_0) \quad 0 < x < \infty \\ \forall_i \quad P_i(0) &= \begin{cases} 1 & x_0 = 0 \text{ and } i = 1 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \quad (3)$$

We assume that the input parameters (i.e., λ, c_A, μ, c_S, λ_i, a_i) are constant on the interval of interest, which is valid because δ is small enough compared to Δ. Then the Laplace transform $f^*(x, s|x_0)$ of $f(x, t|x_0)$ can be computed in terms of Laplace transform of the holding time distribution, denoted by $h^*(s)$. The Stehfest [31] algorithm is used to invert the transform. TSS computational procedure for a single communication link is presented in Figure 5. Details are available in [17].

4 TSS for a Network

The previous section described timestep stochastic simulation of a single link driven by a single flow. We now extend this to a network of queues driven by a set of flows. Each flow represents the traffic generated by a UDP or TCP source, and is defined by the following quantities: source node; sink node; path through the network taken by its packets; and the first two moments of the source's inter-packet generation time during each timestep. (The second moment corresponds to how much jitter the source exhibits with respect to its mean rate.)

initialize internal state of the traffic source;
initialize queue size $N(0)$;
for $t = 0$ **to** $StopTime$ **with step** δ **do**
  obtain moments of the arrival process $\lambda(t)$ and $c_A(t)$;
  obtain probability distribution $Pr[N(t+\delta)|N(t)]$
    using diffusion approximation (Eqns. 1, 2, and 3);
  choose new queue size $N(t+\delta)$ according to Pr;
  write metrics for time t to the file;
end for;

Figure 5: TSS computational procedure for a single communication link and one traffic flow.

The model, of course, allows the source's moments to vary over different timesteps. For a time-dependent source, the moments would depend only on time (e.g., a start and stop time). For a state-dependent source, the moments would depend on the evolution of the network state thus far; for example, a TCP source's rate would depend on the recent history of the sizes of the queues along its path. A model of TCP-Tahoe is discussed in simulation studies in Section 5. We also implemented and evaluated a state-dependent UDP source, but we do not report results here because of the lack of space.

4.1 Departure Process, Splitting and Merging Processes

To extend TSS to a network of queues, we have to be able to compute the first two moments of the departure process of a queue as well as that of processes obtained by splitting and merging flows. We formulate equations for each of these cases next. Throughout, we assume that the first two moments of the arrival and service processes on each link are constant within each δ interval. This assumption is valid because we set δ to be smaller than the control time scale Δ.

Following [6, 5, 34] we approximate the transient departure process using the Marshall's formula [22] with π_0 replaced with $\pi_0(t)$ as computed by the diffusion approximation. For inter-departure time $\phi(t)$, we obtain:

$$\begin{aligned} E[\phi(t)] &= \frac{1}{\mu} + \pi_0(t)E[h] \\ E[\phi(t)^2] &= V_s + \frac{1}{\mu^2} + \pi_0(t)\{E[h^2] + 2\frac{E[h]}{\mu}\} \end{aligned} \quad (4)$$

where V_s is the variance of the service process, μ is the

mean service rate, and $E[h]$ and $E[h^2]$ are first two moments of the idle period.

At a node of a network, the incoming traffic flow may be split into multiple flows (depending on the routing probabilities). Let the inter-arrival time of the process being split have mean rate λ_a and squared coefficient of variation c_a. The packets may enter one of n communication links with probabilities $q_1, q_2, \ldots, q_n$. Let the inter-arrival time of the ith resulting process have mean rate λ_i and squared coefficient of variation c_i. Following [13] and [19], the transition rates and squared coefficients of variation of the ith component process are given by:

$$\begin{array}{l} \lambda_i = \lambda_a * q_i \\ c_i = q_i c_a + (1 - q_i) \end{array} \quad (5)$$

Consider the merge of independent renewal processes. It turns out that the first two moments of the merged process are *not* obtainable from just the first two moments of each of the constituent renewal processes. One has to work with the distributions of the inter-arrival times. We use the approach described in [19], which substitutes a hypo- or hyper-exponential distribution (depending on variances of processes being merged) leading to a closed-form approximate expression. Details are omitted for space reasons (available in [19]).

4.2 Network Flows

TSS computes a sample path of the queue size evolution for each communication link in the network. In order to complete the description of the system and to model congestion control schemes, we need to specify for each communication link the number of packets of each flow that traversed the link in each timestep.

For each flow j and communication link i we introduce a series of flow-conservation equations. Denote by $N_i^j(t)$ the number of packets of flow j in queue i at time t, by $D_i^j(t)$ the number of packets of flow j departing from queue i during $[t, t+\delta]$, by $\lambda_i^j(t)$ mean arrival rate of packets of flow j to queue i during $[t, t+\delta]$, and by $A_i^j(t)$ arrival count of packets of flow j to link i during $[t, t+\delta]$. Under the assumption that the routing is properly configured and flows do not have cycles, we have:

$$A_i^j(t) = \begin{cases} \lambda_i^j(t) * \delta & \text{if } i \text{ is the first-hop of flow } j \\ D_k^j(t) & \text{otherwise, where } k \text{ is the previous link on the path of flow} j \end{cases} \quad (6)$$

Denote by $A_i(t) = \sum_k A_i^k(t)$ the total arrival count to the link i during $[t, t+\delta]$, and by $P_i^{full}(t)$ the probability that a packet arriving during $[t, t+\delta]$ finds the queue full. $P_i^{full}(t)$ is computed using the diffusion equation. Total number of packets lost due to the overflow of queue i within time interval $[t, t+\delta]$, denoted by $L_i(t)$, is determined as:

$$L_i(t) = A_i(t) P_i^{full}(t) \quad (7)$$

Losses are assigned to flows proportionally to the share of a flow in the aggregate traffic. The number of packets lost by flow j on link i during $[t, t+\delta]$, denoted by $L_i^j(t)$, is:

$$L_i^j(t) = L_i(t) \frac{A_i^j(t)}{A_i(t)} \quad (8)$$

Denote the mean arrival rate and squared coefficient of variation of interarrival times to link i during $[t, t+\delta]$ by $\lambda_i(t)$ and $c_{A,i}(t)$, respectively. These values are computed using techniques presented at the beginning of this section. Mean service rate and squared coefficient of variation of service time of queue i during $[t, t+\delta]$ are denoted by $\mu_i(t)$ and $c_{S,i}(t)$, respectively. Queue size $N(t+\delta)$ is randomly chosen according to the probability distribution computed as described in Section 3 based on $N(t)$, $\lambda_i(t)$, $c_{A,i}(t)$, $\mu_i(t)$, and $c_{S,i}(t)$. Departure count of flow j from link i within $[t, t+\delta]$ is determined according to:

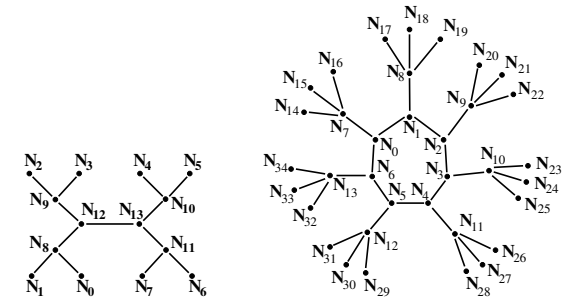
$$D_i^j(t) = N_i^j(t) - N_i^j(t+\delta) + A_i^j(t) - L_i^j(t) \quad (9)$$

5 TCP Network Simulations

We verified the quality of TSS approximation by comparing its results with that of packet level simulations. We studied a broad variety of network topologies with various traffic flow configurations with both TCP and UDP traffic. The TSS achieves high accuracy while computing the results orders of magnitude faster than packet level simulator (several thousand times faster for topologies with 1Gbit links). Due to the lack of space we report on only two example network topologies.

5.1 Model of the TCP Source

We use TCP sources with slow-start, congestion avoidance, and retransmissions driven by timeout (i.e., TCP-Tahoe sources). To represent a TCP-Tahoe source in TSS, we augment the state of a TCP source with the following: (1) congestion window size; (2) slow start threshold; (3) last acknowledged sequence number; (4) RTO timer value, i.e., time before which the last sent data segment has to be acknowledged or loss will be assumed. We also extend the information maintained at each queue for a TCP flow. In addition to the number of packets, TSS also keeps track of the lowest and highest sequence numbers, and updates them based on the arrival, departure, and loss count for the last hop of the return path. The data arrival count and variability



(a) 14-node network topology (b) Large network topology

Figure 6: 14-node network topology with 40 TCP flows (a) and 35-node network topology with 180 TCP flows (b).

at the TCP sink is used by the TCP source as representing the acknowledgment stream. We make the assumption that the acknowledgments encounter a fixed delay.

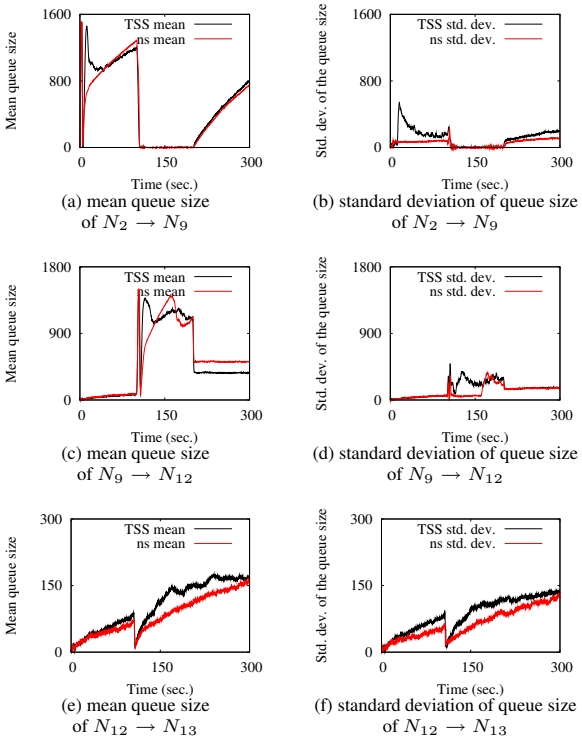
The source adjusts its congestion window based on the statistics of the arriving acknowledgment stream. In particular, if the congestion window is smaller than the slow-start threshold, the congestion window gets increased by the number of received acknowledgments. Otherwise, the source is in congestion avoidance mode and the congestion window is expanded by the number of arriving acknowledgments divided by the current size of the congestion window.

5.2 14-node Network

Our first example configuration consists of 14 nodes and 13 communication links (as shown in Figure 6a). The central link ($N_{12} \rightarrow N_{13}$) has service rate of 2000 packets/sec and all other links have service rates of 1000 packets/sec. Buffering capacity of all links is 1500 packets and squared coefficient of variation of service time is 0.05. Traffic is generated by 40 TCP flows with time-dependent start and stop times. In particular, 10 TCP flows from node N_0 to N_4 and 10 TCP flows from node N_2 to N_6 are active from the beginning of the simulation until 300 seconds. Moreover, 10 TCP flows from node N_1 to N_5 and 10 TCP flows from node N_3 to N_7 are active from 100 seconds until 200 seconds. Figure 7 presents comparison of mean queue size and its deviation as computed by packet-level simulation and TSS for links $N_2 \rightarrow N_9$, $N_9 \rightarrow N_{12}$, and $N_{12} \rightarrow N_{13}$. TSS correctly predicts periods of large queuing for all links of interest. It also approximates variability of the queue size very well.

5.3 35-node Network

We now examine the quality of TSS approximation for a 35-node network with 35 communication links and 180 TCP flows shown in Figure 6b. All links have service rate



(a) mean queue size of $N_2 \rightarrow N_9$ (b) standard deviation of queue size of $N_2 \rightarrow N_9$

(c) mean queue size of $N_9 \rightarrow N_{12}$ (d) standard deviation of queue size of $N_9 \rightarrow N_{12}$

(e) mean queue size of $N_{12} \rightarrow N_{13}$ (f) standard deviation of queue size of $N_{12} \rightarrow N_{13}$

Figure 7: Comparison of mean queue size and its standard deviation as computed by packet-level simulator and TSS for links $N_2 \rightarrow N_9$, $N_9 \rightarrow N_{12}$, and $N_{12} \rightarrow N_{13}$ of network in Figure 6a.

of 5000 packets/sec, which corresponds to (assuming 1KB packets) approximately 40 Mbps. Buffer capacities vary from 1000 to 5000 packets. 10 bulk TCP flows are continuously active for each of the following source-destinations: $N_{14} \rightarrow N_{17}$, $N_{15} \rightarrow N_{18}$, $N_{16} \rightarrow N_{19}$, $N_{14} \rightarrow N_{23}$, $N_{15} \rightarrow N_{24}$, $N_{16} \rightarrow N_{25}$, $N_{20} \rightarrow N_{23}$, $N_{21} \rightarrow N_{24}$, $N_{22} \rightarrow N_{25}$, $N_{20} \rightarrow N_{26}$, $N_{21} \rightarrow N_{27}$, $N_{22} \rightarrow N_{28}$, $N_{29} \rightarrow N_{32}$, $N_{30} \rightarrow N_{33}$, $N_{31} \rightarrow N_{34}$, $N_{29} \rightarrow N_{17}$, $N_{30} \rightarrow N_{18}$, $N_{31} \rightarrow N_{19}$. Because this configuration has multiple interacting TCP flows on a non-trivial topology, it gives raise to various behaviors and is a good configuration for presenting the quality of TSS approximation. Figure 8 presents comparison of mean queue size and its deviation as computed by packet-level simulation and TSS for links $N_0 \rightarrow N_1$, $N_{12} \rightarrow N_5$, and $N_2 \rightarrow N_3$. TSS results match very closely those of packet-level simulation.

6 Related Work

There are many techniques for performance modeling of computer networks (and queuing systems in general). One of the most widely used is packet-level simulation [32, 30], in which an event is simulated for every packet transmission and reception. This technique, if used carefully, can provide

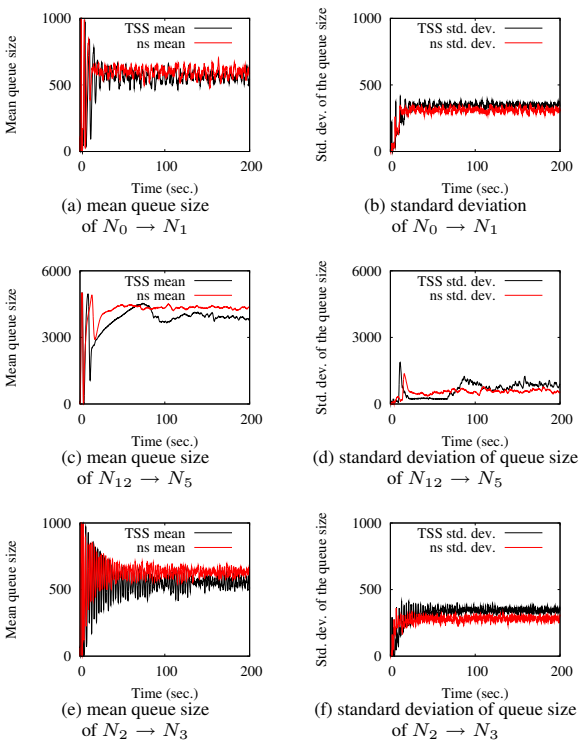


Figure 8: Comparison of mean queue size and its standard deviation as computed by packet-level simulator and TSS for links $N_0 \rightarrow N_1$, $N_{12} \rightarrow N_5$, and $N_2 \rightarrow N_3$ of network in Figure 6b.

very high accuracy. However, it becomes prohibitively expensive for high-speed communication links and large network architectures.

There are many analytical methods to solve for steady-state metrics of queuing networks, for example, [13, 14], and, more relevant to our approach, [34, 33, 9, 15, 16, 10, 19]. There is also work on computation of transient metrics in networks with time-dependent control schemes, for example [6, 5]. However, these methods are not effective for solving networks with complex state-dependent control schemes (such as TCP's congestion control and dynamic adaptive routing).

Fluid approximation methods (e.g., [24, 20, 25, 2, 1, 21]) represent the network by a set of stochastic differential equations, which are then solved in timestep fashion to obtain time evolutions of the network state. So TSS is similar to fluid methods in this respect. However, fluid methods consider the differential equations in only the first moment, whereas TSS considers them in the first two moments. In particular, in one class of fluid methods (e.g., [24, 20, 25]), the set of differential equations is reduced to a set of deterministic differential equations. Consequently, the approach yields only a single evolution, intended to correspond to the ensemble-averaged mean of the metrics. The second group of fluid approximation techniques [2, 1, 21], describe time evolution of TCP congestion window again assuming deterministic network behavior. However, they introduce stochastic description of start and stop times of TCP flows.

There are hybrid approaches that combine stochastic fluid approximations with packet-level simulation, as in [11, 3]. These approaches inherit the limitations of stochastic fluid approximation that makes it difficult to properly capture the effect of state-dependent control causing diverse sample paths.

There are also timestep computation approaches that directly yield the time evolution of an instantaneous ensemble-averaged metric of a network. One example is the Z-iteration [23, 29], which computes instantaneous ensemble-average metrics of interest (e.g., queue size, loss rate, source rate) by approximating the relationship between instantaneous metrics by their steady-state counterparts. But it is restricted to networks of M(t)/M(t)/*/* queues, and so cannot capture the effect of state-dependent control schemes.

We now describe the results in continuous stochastic processes that form the foundation of our TSS approach. There is a large body of literature devoted to diffusion approximations for obtaining queue size distributions. The approaches date back to Kolmogorov [18], who first proposed diffusion equations. Feller [7] extended his ideas and provided the framework for solving various problems using diffusion. In a series of articles devoted to the analysis of road traffic [27, 26, 28], Newell proposed a set of approximation techniques applicable in low, mild, and heavy traffic conditions. Similar work also came from Gaver [8] and Kingman [12].

Gelenbe [9, 10] and Kobayashi [15, 16] proposed the use of diffusion approximations with holding barriers to model queue size evolution for equilibrium and non-equilibrium cases. (There are many types of barriers, including absorbing and reflecting.)

Whitt [34], Duda [6], Kuehn [19] and others have studied the problem of obtaining the statistics of the departure process of a queue, building upon the work by Marshall [22] relating the Laplace transforms of the distributions of service time, idle period, and inter-departure time. Whitt [35] and Kuehn [19] have studied the problem of determining the statistics of traffic flows obtained by splitting and merging other traffic flows in the context of communication networks.

7 Conclusions and Future Work

We presented a novel technique, called timestep stochastic simulation, for fast performance evaluation of computer networks. Our method generates sample paths of the system with very high accuracy. In each step the new state of the system is chosen randomly based on the current state

and a probability distribution obtained using diffusion approximation. The method is much faster than packet-level simulation and has almost the same accuracy. Because state-dependent control gets feedback based on sample path metrics, our method is more suitable for modeling state-dependent control schemes (such as TCP's) than fluid approximations. Our future research plans include extending the TSS with RED queue model. We also plan to apply it to various networking problems, such as dynamic routing, load balancing, or optimal cache placement.

References

[1] F. Baccelli and D. Hong. Flow Level Simulation of Large IP Networks. In *INFOCOM*, 2003.

[2] F. Baccelli and D. Hong. Interaction of TCP Flows as Billiards. In *INFOCOM*, 2003.

[3] S. Bohacek, J. P. Hespanha, J. Lee, and K. Obraczka. A Hybrid Systems Modeling Framework for Fast and Accurate Simulation of Data Communication Networks. In *SIGMETRICS*, 2003.

[4] D. R. Cox. A use of Complex Probabilities in the Theory of Stochastic Processes. In *Cambridge Phil. Soc.*, 1955.

[5] A. Duda. Transient Diffusion Approximation for some Queueing Systems. In *SIGMETRICS*, 1983.

[6] A. Duda. Diffusion Approximation for Time-dependent Queueing Systems. In *IEEE JSAC*, pages 905–918, 1986.

[7] W. Feller. Diffusion Processes in One Dimension. In *Amer. Math. Soc.*, volume 77, pages 1–31, July 1954.

[8] D. P. Gaver. Diffusion Approximations and Models for Certain Congestion Problems. In *Journal of Applied Probability*, volume 5, pages 607–623, 1968.

[9] E. Gelenbe. On Approximate Computer System Models. In *ACM (JACM)*, volume 22, pages 261–269, April 1975.

[10] E. Gelenbe. Probabilistic Models of Computer Systems, Part II: Diffusion Approximation, Waiting Times and Batch Arrivals. *Acta Informatica, vol. 12, pp 285-303*, 1979.

[11] Y. Guo, W. Gong, and D. Towsley. Time-stepped Hybrid Simulation (TSHS) for Large Scale Networks. In *INFOCOM*, 2000.

[12] J. F. C. Kingman. The Heavy Traffic Approximation in the Theory of Queues. In *Symp. on Congestion Theory, Univ. North Carolina Press, Chapel Hill*, pages 137–169, 1965.

[13] L. Kleinrock. *Queueing Systems - volume I*. John Wiley and Sons, 1976.

[14] L. Kleinrock. *Queueing Systems - volume II*. John Wiley and Sons, 1976.

[15] H. Kobayashi. Application of the Diffusion Approximation to Queueing Networks I: Equilibrium Queue Distributions. In *ACM (JACM)*, volume 21, pages 316–328, April 1974.

[16] H. Kobayashi. Application of the Diffusion Approximation to Queueing Networks II: Nonequilibrium Distributions and Applications to Computer Modeling. In *ACM (JACM)*, volume 21, pages 459–469, July 1974.

[17] A. Kochut. *Timestep Stochastic Simulation of Computer Networks using Diffusion Approximation*. PhD Dissertation, University of Maryland, College Park. http://hdl.handle.net/1903/2691, 2005.

[18] A. Kolmogorov. Ueber die Analytischen Methoden in der Wahrscheinlichkeitsrechnung. In *Mathematical Annals*, volume 104, pages 415–458, 1931.

[19] P. Kuehn. Approximate Analysis of a General Queuing Networks by Decomposition. In *IEEE Transactions on Communications*, volume 27, pages 113–126, 1979.

[20] Y. Liu, F. L. Presti, V. Misra, D. Towsley, and Y. Gu. Fluid Models and Solutions for Large-Scale IP Networks. In *SIGMETRICS*, 2003.

[21] M. Marsan, M. Garetto, P. Giaccone, E. Leonardi, E. Schiattarella, and A. Tarello. Using Partial Differential Equations to Model TCP Mice and Elephants in Large IP Networks. In *INFOCOM*, 2004.

[22] K. T. Marshall. Some Relationships Between the Distributions of Waiting Time, Idle Time and Interoutput Time in the $GI/G/1$ Queue. *SIAM Jour. of Appl. Math.*, 16, 1968.

[23] I. Matta and A. U. Shankar. Z-Iteration: A Simple Method for Throughput Estimation in Time-Dependent Multi-Class Systems. In *Measurement and Modeling of Computer Systems*, pages 126–135, 1995.

[24] V. Misra, W. Gong, and D. Towsley. Stochastic Differential Equation Modeling and Analysis of TCP-Windowsize Behavior. In *SIGMETRICS*, 1999.

[25] V. Misra, W. Gong, and D. Towsley. Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED. In *SIGCOMM*, 2000.

[26] G. Newell. Queues with time-dependent arrival rates II: The Maximum Queue and the Return to Equilibrium. In *Journal of Applied Probability*, volume 22, pages 579–590, 1968.

[27] G. F. Newell. Queues with Time-Dependent Arrival Rates: I - The Transition Through Saturation. In *Journal of Applied Probability*, volume 5, pages 436–451, 1968.

[28] G. F. Newell. Queues with time-dependent arrival rates III: The mild rush hour. In *Journal of Applied Probability*, volume 5, pages 591–606, 1968.

[29] C. T. Popescu and A. U. Shankar. Empirical TCP Profiles and Applications. *ICNP*, 1999.

[30] Scalable Network Technologies. Qualnet. http://www.scalable-networks.com.

[31] H. Stehfest. Algorithm 368. Numerical inversion of Laplace transforms. In *Communications of the ACM*, volume 13, pages 47–49, 1970.

[32] University of Southern California. The Network Simulator - ns-2. http://www.isi.edu/nsnam/ns.

[33] W. Whitt. Performance of the Queueing Network Analyzer. In *Bell System Technical Journal*, volume 62, 1983.

[34] W. Whitt. The Queueing Network Analyzer. In *Bell System Technical Journal*, volume 62, 1983.

[35] W. Whitt. Approximations for Departure Processes and Queues in Series. In *Naval Research Logistics Quarterly*, volume 31, pages 499–521, December 1984.