ABSTRACT

Title of Dissertation: Timestep Stochastic Simulation of Computer Networks using Diffusion Approximation

Andrzej Kochut, Doctor of Philosophy, 2005

Dissertation directed by: Prof. A. Udaya Shankar
Department of Computer Science

Performance evaluation of modern computer networks is challenging because of their large sizes, high speeds of communication links, and complex state-dependent control mechanisms. In particular, TCP congestion control reacts in a nonlinear fashion to the state of the network at the time scale of round-trip times, making analysis intractable. Thus packet-level simulation is the only widely used method of performance evaluation. Although it can be accurate, it is computationally expensive and thus can be applied only to small networks and low link speeds.

Timestep Stochastic Simulation (TSS) is a novel method for generating sample paths of computer networks, in increments of time steps rather than packet transmissions. TSS has a low computation cost independent of packet rates and provides adequate accuracy for evaluating general state-dependent control mechanisms. 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, it can accurately model state-dependent control mechanisms, including TCP congestion control, adaptive dynamic routing, and so on.

We have a TSS implementation for general computer networks with state-dependent control. We have applied this to numerous networks with TCP and state-dependent UDP flows, and confirmed its accuracy against packet-level simulation.

Timestep Stochastic Simulation of Computer Networks using Diffusion Approximation

by

Andrzej Kochut

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2005

DEDICATION

To my beloved wife Renata and my wonderful parents

who are always there to support me.

ACKNOWLEDGEMENTS

I wish to thank my advisor, professor A. Udaya Shankar, for his guidance and support during my time at Maryland. He led me through challenging research and provided encouragement during the tough times. He was always there for me to discuss the ideas and answer my questions. The discussions we had were not only a tremendous help, but also taught me how to approach difficult problems and how to be persistent.

I would also like to thank professors Ashok Agrawala and Samrat Bhattacharjee for their encouragement and help. I am grateful to my fellow students, especially Arunchandar Vasan, Christopher Kommareddy, and Bao Trinh. I owe them a great deal for making my time here at Maryland a joyful one.

Internships at IBM T.J. Watson Research Center provided me with great experience. I would especially like to thank Kirk Beaty, Norman Bobroff, and Gautam Kar for the research guidance and continued support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Modern computer networks employ packet switching with various kinds of dynamic control mechanisms, including end-to-end congestion control, and adaptive routing. This approach makes the network very flexible and expandable; large networks can consist of hundreds of thousands of nodes and links. It also makes the network capable of serving as a transportation medium for broad variety of services ranging from simple bulk data transfers to voice and video transmissions. However, the dynamic control mechanisms make the network very difficult analyze and to manage. Yet performance evaluation is crucial both for research and development of new computer networks as well as for management and growth of existing computer networks. In research, for example, whenever a new control mechanism (e.g., a new web caching scheme) is proposed, its performance has to be compared against existing control mechanisms before it can be considered a viable alternative. In network management, for example, internet service providers typically have “Service Level Agreements” with their customers, defining the expected level of service as well as penalties for failing.

Although performance evaluation of computer networks is crucial, existing evaluation techniques are not adequate for handling the large heterogeneous architectures and state-dependent control schemes that are intrinsic to modern computer networks. Currently, the only viable method of performance evaluation of computer networks is **packet-level simulation**, which simulates the computer network at the level of transmissions and receptions of individual packets.

Packet-level simulation can be very accurate, but it is computationally very expensive, precisely because every packet transmission and reception is explicitly simulated. A recent study [1] of packet-level simulators concludes that about 100,000 packet transmissions can be simulated per second on a PC workstation. Simulating a network of modest size and speed, for example, one with tens of 100 Mbps (megabit per second) links, can take many hours. In contrast, a single 20 Gbps (gigabit per second) link requires a packet-level simulator to handle more than 2 million packet transmissions for each second of simulated time. A single state-of-the-art Cisco 12000 series router [2] supports 16 links each of 20 Gbps bandwidth. A modern network can consist of hundreds or thousands of such links. Clearly, packet-level simulation of such networks is practically unattainable. One can attempt to parallelize the simulation over many computers, but this is very expensive and technically exceedingly difficult [3].

Purely analytical techniques (e.g. [4, 5]) do not capture the effects of state-dependent control or realistic traffic mix with reasonable accuracy. They make so many simplifying assumptions (e.g., stationary memoryless arrivals to each link) that the estimates they provide are too inaccurate to be of use.

This current state of affairs has motivated techniques, such as fluid approximation (e.g. [6, 7, 8]), that can handle time-varying arrival and service distributions and yield a single time evolution of the system, which is intended to be representative of (most or all) evolutions of the system (usually in some ensemble-averaged sense). However, these techniques do not yield sample paths or the time evolution of metrics along sample paths, which is exactly what control schemes base their decisions on. *Consequently, these techniques are restricted to systems in which most sample paths are "close" to each other so that the ensemble of sample paths can be inferred from a single "representative" path.* This is often the case when control is time-dependent but not state-dependent, e.g., control based on time of day. But it is usually not the case when the time dependency is caused by state-dependent control, especially the nonlinear kind of state-dependent control present in most computer networks. Here, small changes in observed state result in small changes in behavior over *small time scales*, but lead to large changes and diverse evolutions over *large time scales*. For example, a small difference in link cost causes a change in next hop, which then gradually affects traffic flows throughout the network; or a small difference in roundtrip time causes synchronization between TCP flows sharing a bottleneck.

1.1 Diversity of sample paths

To explain this diversity of sample paths in more quantitative terms, consider a system such as a network of TCP flows. Let $S(t)$ denote the state of the system at time t (e.g., router queue sizes, source window sizes, and so on). Let the incremental evolution of the system be governed by an operator F (defined by the control algorithms, queuing disciplines, network topology, etc.). So the evolution of $S(t)$ is governed by $S(t + \Delta) = F(S(t))$ for some small Δ (for example, TCP roundtrip times).

In general, $S(t)$ includes some random elements and F is a stochastic operator. Thus the system has a set of possible evolutions (or sample paths). Each evolution

x satisfies $S_x(t + \Delta) = F_x(S_x(t))$, where F_x is F instantiated for the particular sequence of random choices corresponding to x. Consider the set of possible evolutions starting from a given state $S_x(t)$ at time t. At time $t+\delta$ for $\delta < \Delta$, the possible states $S(t+\delta)$ are clustered close to each other, and so the evolution over the interval $[t, t + \delta]$ may be reasonably well characterized by a single evolution $S(t + \delta) = \bar{F}(S(t))$, where $\bar{F}$ is F with stochastic operators replaced by their expectations. However, over time scales much larger than Δ, the sample paths will exhibit great diversity and would not be characterized by the single evolution generated by $S(t + \delta) = \bar{F}(S(t))$.

Averaging over the ensemble $\{S_x(t)\}$ of evolutions yields the time evolution of the instantaneous expected state, $E[S(t)]$. This characterizes the ensemble of evolutions quite well and is something we are very much interested in. However, it is very unlikely that $E[S(t)]$ satisfies the incremental evolution $E[S(t + \Delta)] = F(E[S(t)])$ or $E[S(t+\Delta)] = \bar{F}(E[S(t)])$. In the same way, for a metric $M(t)$ (say the expectation of a component of $S(t)$), it is very unlikely that the evolution of $M(t)$ satisfies $M(t + \Delta) = F(M(t))$ or $E[M(t + \Delta)] = \bar{F}(M(t))$. But this is exactly what is needed if the single "representative" evolution computed in the above mentioned techniques is to be truly representative.

This discussion is illustrated in Figure 1.1, where a small difference in the system state at time T can result in two very different future evolutions. The congestion control adapts by reducing sending rate in case when $S(T) > s_0$, forcing majority of the sample paths with this property to have low values of $S(t)$ for $t > T$. On the contrary, most of the sample paths for which $S(T) \leq s_0$ have high values of $S(t)$ for $t > T$. Fluid approximation will follow one of the two paths (depending on the value of $S(T)$ as computed by fluid approximation).

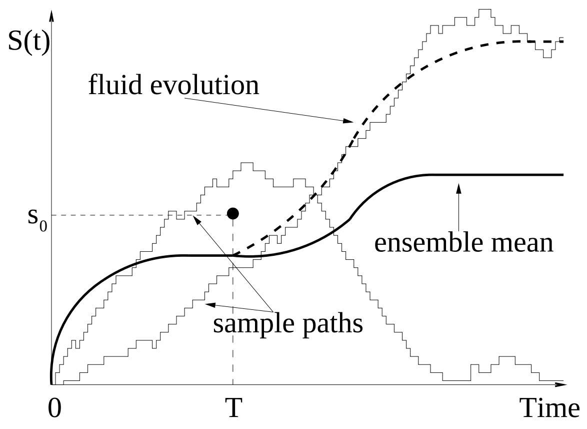


Figure 1.1: Two sample paths and time evolutions of ensemble queue size, and ensemble queue size as computed by fluid approximation

The correct ensemble however is between the two extreme behaviors.

Diversity like that is common even in small TCP/IP networks, such as the one depicted on Figure 1.2. The network consists of two communication links both having service rate of 16Mbs and buffers of 1.6MB. Due to the service time variability queue size process exhibits wide diversity on the timescale of hundreds of seconds.

1.2 Timestep stochastic simulation

We propose a novel method, called **Timestep Stochastic Simulation (TSS)**, that combines discrete-event simulation and analytical approximations to yield *sample paths* of high accuracy and low computational cost (independent of packet rates and queue sizes). Our method computes the evolution of $S(t)$ on a sample path at time instants t_0, t_1, t_2, $\cdots$, where $t_{i+1} - t_i = \delta$ for some $\delta < \Delta$ and all i, assuming a starting state $S(t_0)$.

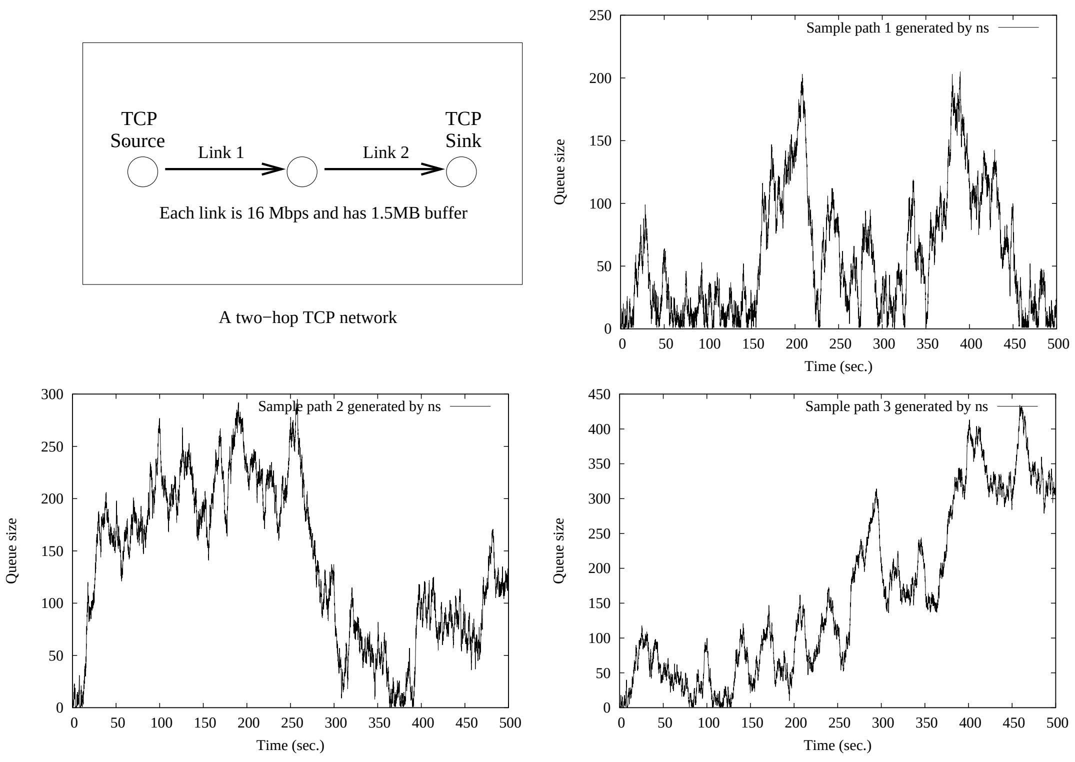


Figure 1.2: A TCP network (upper left) and three sample paths of queue size of link 2.

The idea is to do the following for $i = 0, 1, \cdots$:

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

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

Repeating these two steps for successive intervals generates a sample path. Figure 1.3 depicts this procedure.

The difficulty, of course, is in obtaining an analytical expression for $Pr[S(t_{i+1})|S(t_i)]$. The arrival and service distributions in the interval $[t_i, t_{i+1}]$ are available from $S(t_i)$ and F because we choose δ to be smaller than the time-scale Δ of the control mechanisms. Even so, exact results for $Pr[S(t_{i+1})|S(t_i)]$ are not known

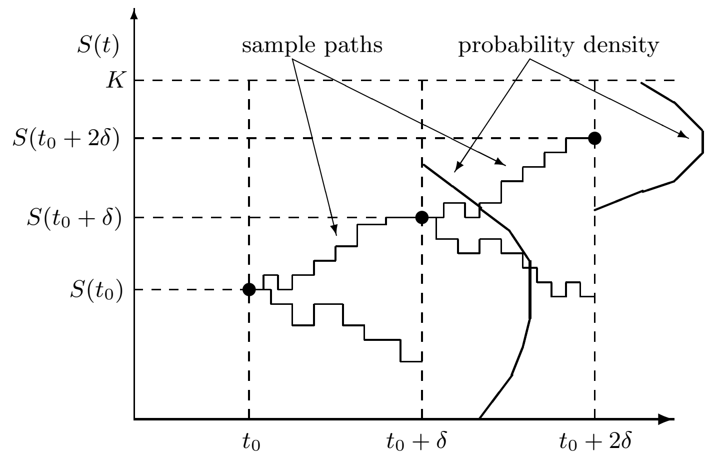


Figure 1.3: Computation procedure. At each step, distribution of the next state is computed and new state randomly chosen based on this distribution.

for most queuing systems of interest. However, the diffusion approximation, first proposed by Kolmogorov [9] and later extended by Feller [10], proves to be excellent at obtaining a very accurate approximation of this distribution. So our method generates sample paths of a stochastic process, say $R(t)$, such that for $t_0, t_1, \cdots$ and $t_{i+1} - t_i = \delta < \Delta$,

$$Pr[R(t_{i+1})|R(t_i)] \approx Pr[S(t_{i+1})|S(t_i)]$$

We refer to $R(t)$ as a δ-*timestep* version of $S(t)$.

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 provides the capability for performance evaluation of “large-scale” problems, i.e., which make sense only in the context of large networks and/or high-speed links. One example of such a problem is the long-standing issue of how aggressive TCP (or any end-to-end) congestion control can become and yet not have its widespread deployment bring down a large network. Another example is the efficacy of access and routing control mechanisms in providing QoS in large ISP networks.

1.3 Contributions

TSS achieves stochastic simulation of computer networks at the level of timesteps, rather than at the level of packet transmissions and receptions. That is, given the state of the network at time t, TSS computes the probability distribution of the network state at time $t + \delta$ and chooses the state randomly according to the distribution. So the computational cost of each step is independent of the link bandwidth, whereas in packet-level simulation, increasing link speed increases both the computational cost (increased number of packet transmissions per second of simulated time) and the memory cost (larger packet queues and event lists).

Because TSS generates sample paths, it is suitable for handling networks with general state-dependent control, unlike deterministic flow-level simulation approaches, such as fluid approximation.

It is typical for computer networks to have communication links with utilization close to 1 for extended periods of time. For example, consider a TCP flow over a sequence of equal-speed links, and suppose that TCP source is capable of saturating the first hop. TCP, after a brief slow-start period enters congestion avoidance mode in which the first link is nearly constantly busy. Consequently, the second-hop link on the path has utilization close to 1 for prolonged periods of time. The queuing process of such a link exhibits very diverse behavior on large timescales, thus strongly affecting delay on the path and the dynamics of the connection. Situation like this is very well handled by TSS, and it is very difficult to model using other approaches, such as fluid approximation.

Another advantage of our approach is the ability to generate not only the mean evolution of the queue size, but also higher order moments. This is impossible to obtain by any other means than the packet-level simulation, which, as we discussed before, is computationally expensive.

Traffic source model in TSS is a time-stepped version of the packet-level model. It is quite easy and intuitive to develop since it operates in terms of sample executions and not ensembles as is the case in, for example, fluid approximations. We illustrate the method with three source models: time-dependent UDP, state-dependent UDP, and TCP.

Finally, we evaluate the accuracy of TSS by extensive comparison against packet-level simulations.

1.4 Organization of the Dissertation

The remainder of the Dissertation is organized as follows. Chapter 2 reviews related work, including the results in stochastic processes on which our approach is based. Chapter 3 presents the details of timestep stochastic simulation for one communication link. Chapter 4 describes the extension of the method to networks, in particular, obtaining the first two moments of queue departure processes and of processes resulting from merging and splitting processes. Chapter 5 describes how to design traffic flow models in the TSS. It also presents three examples of traffic sources: time-dependent UDP flow, state-dependent UDP flow, and TCP. Chapter 6 presents results of numerical evaluation of TSS. The results reported span multiple network architectures. Chapter 7 summarizes contributions and outlines possible directions of future research.

Chapter 2

Related Work

Modeling and performance evaluation of computer networks is a broad research area that builds upon techniques and methods from diverse fields, including discrete-event simulation of computer systems, steady-state analysis of manufacturing processes, control systems theory, queuing theory, and road traffic analysis and management.

We can divide the related work into several major categories. They include packet-level simulation, analytical modeling, approximation techniques, numerical methods used in modeling, and congestion control schemes. The remainder of this chapter reviews these categories, outlines their advantages and problems, and contrasts with TSS method.

2.1 Packet-level simulation

One of the most widely used techniques for performance modeling of computer networks is packet-level simulation [11, 3], in which an event is simulated for every packet transmission and reception. This technique, if used carefully, can provide very high accuracy. However, it becomes prohibitively expensive for high-speed communication links and large network architectures. In particular, increasing the bandwidth of communication links causes an explosion in the number of events that need to be handled by the simulator. For example, a 10 Mbps link can handle approximately 1200 1KB-packets per second, which results in 2400 arrival and departure events. However, a modern fast communication link (such as at the ports of a Cisco 12000 series router [2]) can have a speed of 20 Gbps, resulting in a few million events per second. A study of currently available simulation tools [1] concludes that the state-of-the-art simulators on a single PC can handle less than 100,000 of packet transmissions per second corresponding to achieving real-time simulation for at most a single 1 Gbps link.

2.2 Analytical methods

There are many analytical methods to solve for steady-state metrics of queuing networks, for example, [4, 5, 12, 13, 14]. Most notably, it is possible to solve Jacksonian-type networks, where external arrivals have stationary Poison distributions, service times are Markovian, and only random Bernoulli routing is allowed. However, more general networks do not have analytical solutions. For example, there are no known solutions even for the mean queue sizes of networks of queues where either service or arrival processes are not Markovian. Time-dependent traffic sources are even more difficult to treat analytically. The newest solutions [15] for the simple case of one queue with time-dependent exponential arrival and service do not generalize to network of queues. Modern computer networks present even bigger challenges due to the predominance of dynamic state-dependent control. There are no analytical methods that can address problem of obtaining even the simplest statistics of such systems.

2.3 Computational analysis methods

The prohibitive cost of packet-level simulation and the infeasibility of analytical models has sparked significant research on developing approximation solutions that would be applicable to a broader range of systems than that covered by classical queuing theory. One such solution is Queuing Network Analyzer [16, 17]. It is applicable to steady-state analysis of network of queues with general but stationary arrival and service processes. It approximates all traffic flows using first two moments of a renewal process that matches best the original process. The network is decomposed into a series of queues, and the mean size of each of them is computed separately. Mean queue sizes are obtained using Kraemer and Langenbach-Beltz approximation [18]. Several improvements to the method were developed that address issues related to correlation of internal network flows [19, 20], arising from merging and splitting of traffic flows as well as deterministic routing [21]. Another decomposition-based approximation method was developed by Kobayashi [22, 23]. It solves for mean steady-state queue sizes in network of queues also assuming that all traffic flows and service are renewal i.i.d. processes characterized by first two moments of their distributions. Diffusion approximation is used to obtain mean queue size based on the first two moments of arrival and service processes. Similar approaches are also used in [24, 25] and in [26].

2.4 Mean value analysis of time-dependent systems

There is also work on computation of transient metrics in networks with time-dependent control schemes, for example [27, 28], which extends the approach of [24, 25]. 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). Traffic sources in this model may adjust their sending rate and variability, however not based on the network conditions (such as round-trip time or loss rate) but only on time (for example source can double its rate at 10th second of evolution). This constraint makes it impossible to extend the method to state-dependent control schemes such as TCP.

There are also numerical-analytical approaches that yield the time evolution of the instantaneous ensemble-averaged metric of a network. One example is the Z-iteration [29, 30], 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. It achieves high accuracy and low computational cost for networks of M(t)/M(t)/*/* queues. But it only computes a single evolution and so, as described in the Introduction, cannot capture the effect of state-dependent control schemes that induce diversity in sample paths.

Stochastic fluid approximation [6, 7, 31] is another approach that yields a single time evolution of an instantaneous ensemble-averaged metric of a network. This approach treats a packet flow as a fluid and represents the system by a set of stochastic differential equations. However, the solution procedure considers the differential equations only in the expectation, i.e., as a set of deterministic differential equations. Consequently, the stochastic fluid approach solves only for a single evolution, corresponding to the ensemble-averaged mean of the metrics. Thus it too has the limitations described in the introduction regarding capturing the effects of state-dependent control that induces diversity in sample paths.

There are hybrid approaches that combine stochastic fluid approximations with packet-level simulation, as in [8, 32]. Parts of the network are simulated using packet-level simulation, while rest of the network is modeled using fluid-based differential equations. 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.

2.5 Diffusion approximation

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 [9], who first proposed diffusion equations. Feller [10] 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 [33, 34, 35], Newell proposed a set of approximation techniques applicable in low, mild, and heavy traffic conditions. Similar work also came from Gaver [36] and Kingman [37].

Gelenbe [24, 25] and Kobayashi [22, 23] proposed the use of diffusion approximations with *holding barriers* to model queue size evolution for equilibrium and non-equilibrium cases. (There are many other types of barriers, including absorbing and reflecting.)

Whitt [16], Duda [27], Kuehn [26] and others have studied the problem of obtaining the statistics of the departure process of a queue, building upon the work by Marshall [38] relating the Laplace transforms of the distributions of service time, idle period, and inter-departure time. Whitt [39], Fraker [40], and

Kuehn [26] 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. Their approach for merging multiple flows was further refined by Albin [19]. She proposed a convex combination of stationary-interval and asymptotic approximations that minimizes the approximation error.

Using the diffusion approximation with holding barriers to obtain transient queue size distribution requires one to work with Laplace transforms. There are no closed-form solutions in the time domain, so numerical inversion of the transform is required. We use Stehfest algorithm [41] for inversion. It numerically inverts the transform computing its values only at real points. This approach is fast, but may suffer from low accuracy.

2.6 Congestion control

TSS is designed to model computer networks with state-dependent congestion control. There is a broad variety of protocols for managing data transmission in IP networks. UDP (User Datagram Protocol) [42] supports data transmission without congestion control. A UDP source sends data into the network without any intrinsic constraints; it is restricted only by the application using UDP. More interesting to model is TCP (Transmission Control Protocol) [43]. TCP uses a window-based scheme to determine a data rate that results in efficient transport while simultaneously protecting the network from permanent congestion. Since its introduction in 1981, TCP has been significantly improved and extended. Most notably, the introduction of slow-start, congestion avoidance, fast retransmit, and fast recovery algorithms [44] has improved the protocol performance. There are additional improvements (e.g., [45, 46]).

Chapter 3

Timestep Stochastic Simulation for One Queue

This chapter develops the technique of timestep stochastic simulation for a single communication link. We model the communication link by a single-server queue in the usual way, with service rate equal to the link bandwidth (in packets/second), squared coefficient of variation capturing variability of service time, 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 $N(t)$.

Recall that TSS computes a sample path in timesteps of size δ. We make the following assumptions:

- Statistics of both inter-arrival and service time processes are described using first two moments of their distributions and remain constant within each timestep

- Mean arrival rate is large enough so that within each timestep of size δ, the number of arriving customers is sufficiently large for the normal approximation to hold (i.e., not less than 30)

- Congestion control mechanisms, which adjust the external arrival rates and

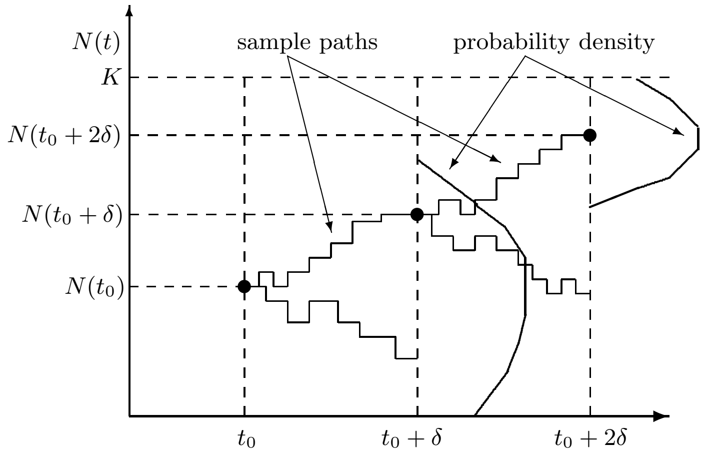


Figure 3.1: Computation procedure. At each step, distribution of the queue size at next step is computed and new queue size randomly chosen based on this distribution.

variability, make their adjustments in intervals Δ, where $\delta < \Delta$, and thus the external arrival rates and variability is constant within each timestep

3.1 Overview

We compute a time-stepped evolution of the stochastic process $N(t)$ given a starting state $N(t_0)$ as illustrated in Figure 3.1 (which is the same as Figure 1.3 with S replaced by N). 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 value for $N(t_{i+1})$ with this distribution.

TSS computational procedure for a single communication link is presented on Figure 3.2. First, internal state of the arrival flow is initialized. This state may be as simple as a fixed values of arrival rate λ and squared coefficient of variation c_A that remain constant throughout the entire simulation in case of constant-rate UDP source, or as complex as the congestion control state in case

```
initialize internal state of the traffic source;
initialize queue size $N(0)$;
for $t = 0$ to $StopTime$ with step $\delta$ do
    compute moments of the arrival process $\lambda(t)$ and $c_A(t)$;
    obtain probability distribution $Pr[N(t+\delta)|N(t)]$;
    choose new queue size $N(t+\delta)$ according to distribution $Pr$;
    dump metrics for time $t$;
end for;
```

Figure 3.2: TSS computational procedure for a single communication link and one traffic flow.

of a TCP source. Next, the queue size $N(0)$ is initialized. The remainder of the *TSS* simulation proceeds in steps of size δ. In each step, the first two moments of the arrival process are determined based on the internal state of the traffic source as well as, in the case of TCP, on the queue size (including its history). Then new queue size $N(t+\delta)$ is chosen based on the distribution $Pr[N(t+\delta)|N(t)]$. It is obtained by inversion of Laplace transforms shown on Equations 3.14, 3.15, and 3.18 using numerical procedure described in Section 3.4. Finally, the metrics of interest are written to the file.

Obtaining the transient distribution $Pr[N(t_{i+1})|N(t_i)]$ for a general arrival and service processes is non-trivial. Exact solutions exist only for special cases, including, most notably, recent work for M(t)/M(t)/1 queue by Knessl and Yang [15]. However, exact results are not known for the majority of queues of interest, for example, single-server queues with high-variance arrival distributions and low-variance service distributions. Thus we have to resort to approximate techniques to obtain the distribution of interest.

Our choice is to use the diffusion approximation introduced by Kolmogorov [9] and further refined by Feller [10]. In this approximation, the queue size stochastic process $N(t)$ is approximated by a continuous Wiener stochastic process. The arrival and service processes can each have a time-varying distribution characterized by the first two instantaneous moments. The Wiener process $N(t)$ is subject to a lower holding barrier at $N(t) = 0$ (corresponding to an empty queue) and an upper holding barrier at $N(t) = K$ (corresponding to a full queue). Whenever $N(t)$ reaches the lower (or upper) barrier, it stays there for a random duration with a *holding time* distribution and then returns to the point $N(t) = 1$ (or $N(t) = K-1$). [The theory allows for a random return point in the open interval $(0, K)$ with arbitrary distribution.]

The holding time distributions at the barriers depend on the arrival and service distributions. If arrivals are exponential, the holding time at the lower barrier is exponentially distributed with the same rate as that of arrivals. Similarly, if service is exponential, the holding time at the upper barrier is exponentially distributed with the same rate as the service. For other arrival and service processes, Gelenbe [25] proposed representing holding time distribution by the Coxian distributions [47]. Coxian distribution can fit an arbitrary number of moments of any general distribution (if the parameters of the Coxian distribution are allowed to take complex values [47]).

The following subsection introduces the diffusion equation with two holding barriers.

3.2 Diffusion processes in one dimension

We start by informally introducing some concepts from the theory of diffusion processes. Our treatment follows [48]. For more formal treatment see [10, 49].

Consider a particle moving on the real line initially positioned at the origin. In each small time interval $\Delta t > 0$, the particle changes its position by $+\Delta x$ or

$-\Delta x$ for some $\Delta x > 0$. Denote the magnitude of change by a random variable Z. Let the probability of a move in the positive direction be p and in the negative direction $q = 1 - p$. Thus:

$$\begin{aligned} Pr[Z = +\Delta x] &= p \\ Pr[Z = -\Delta x] &= q \end{aligned} \tag{3.1}$$

Assume also that the steps are mutually independent. Moment generating function of one transition $m_Z(z)$ is given by:

$$m_Z(z) = pe^{-z\Delta x} + qe^{z\Delta x} \tag{3.2}$$

Let $X(t)$ denote the position of the particle at time t. Since the transitions are mutually independent and identically distributed, the moment generating function $m_{X(t)}(z)$ of the total displacement $X(t)$ of the particle at time t may be computed by multiplying $m_Z(z)$ with itself $\frac{t}{\Delta t}$ times.

$$m_{X(t)}(z) = [m_Z(z)]^{\frac{t}{\Delta t}} = \left[pe^{-z\Delta x} + qe^{z\Delta x}\right]^{\frac{t}{\Delta t}} \tag{3.3}$$

The mean $E[X(t)]$ and variance $V[X(t)]$ of the displacement at time t may be easily computed from this:

$$\begin{aligned} E[X(t)] &= \tfrac{\Delta x}{\Delta t}(p - q)t \\ V[X(t)] &= 4\tfrac{\Delta x^2}{\Delta t}pqt \end{aligned} \tag{3.4}$$

Consider the limiting case of $\Delta x \to 0$ and $\Delta t \to 0$. To obtain a non-degenerate result with the limiting process having mean β and variance α^2, we need to set the following constraints:

$$\begin{aligned} \Delta x &= \alpha\sqrt{\Delta t} \\ p &= \tfrac{1}{2}\left(1+\tfrac{\beta\sqrt{\Delta t}}{\alpha}\right) \\ q &= \tfrac{1}{2}\left(1-\tfrac{\beta\sqrt{\Delta t}}{\alpha}\right) \end{aligned} \tag{3.5}$$

To derive the differential equation describing the flow of probability in this model, denote by $f(x,t|x_0)\Delta x$ the probability that the particle at time t is in the interval $[x, x+\Delta x]$ assuming that it was at x_0 at time $t=0$. From conservation of probability mass, we have:

$$\begin{aligned} f(x,t|x_0)\Delta x &= pf(x-\Delta x, t-\Delta t|x_0)\Delta x \\ &\quad +qf(x+\Delta x, t-\Delta t|x_0)\Delta x \end{aligned} \tag{3.6}$$

Replacing $f(x-\Delta x, t-\Delta t|x_0)$ and $f(x+\Delta x, t-\Delta t|x_0)$ by their Taylor expansions around (x,t), replacing p, q, and Δx with values from Equation 3.5 and finally going to the limit $\Delta t \to 0$, we obtain the following diffusion differential equation for the open real line (arguments of f are omitted for brevity):

$$\frac{\partial f}{\partial t} = \frac{\alpha^2}{2}\frac{\partial^2 f}{\partial x^2} - \beta\frac{\partial f}{\partial x} \tag{3.7}$$

Further refinement of the unbounded diffusion process is the introduction of barriers, in our case constraining the particle to the interval $[0,K]$. There are several types of barriers: reflecting, holding, and absorbing. For the purposes of TSS we use holding barriers with Coxian holding distributions introduced by Gelenbe [25]. Whenever the particle reaches one of the barriers (located at 0 and K), it is held there for a period of time distributed according to the Coxian distribution. Then it jumps to a random point within the open interval $(0, K)$ distributed according to $f_{upper}(x), x \in (0, K)$ if transiting from the upper barrier and $f_{lower}(x), x \in (0, K)$ if transiting from the lower barrier.

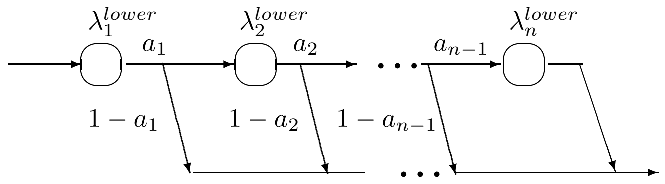


Figure 3.3: Coxian distribution with n stages representing the distribution of lower boundary holding time.

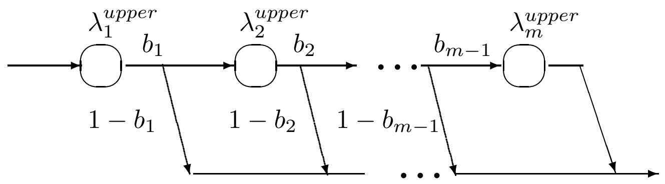


Figure 3.4: Coxian distribution with m stages representing the distribution of upper boundary holding time.

Assume that holding time at the lower barrier is distributed according to n-stage Coxian distribution with i-th stage transition rate equal to λ_i^{lower} and i-th stage escape probability of $1 - a_i$, as presented on Figure 3.3. Similarly, assume that holding time at the upper barrier is distributed according to m-stage Coxian distribution with i-th stage transition rate equal to λ_i^{upper} and i-th stage escape probability of $1 - b_i$, as presented on Figure 3.4. Denote the probability of finding the particle at time t in stage i at the lower boundary by $P_i(t)$ and in stage i

at the upper boundary by $Q_i(t)$. Probability mass conservation Equation 3.6 becomes:

$$\begin{aligned} f(x,t|x_0)\Delta x &= pf(x-\Delta x, t-\Delta t|x_0)\Delta x + qf(x+\Delta x, t-\Delta t|x_0)\Delta x \\ &\quad + \sum_{i=1}^{n} P_i(t) f^{lower}(x) \lambda_i^{lower}(1-a_i)\Delta x \Delta t \\ &\quad + \sum_{i=1}^{m} Q_i(t) f^{upper}(x) \lambda_i^{upper}(1-b_i)\Delta x \Delta t \end{aligned} \tag{3.8}$$

Equation 3.8, after the same limits as in the case of deriving Equation 3.7 from Equation 3.6, leads to (again parameters of f are omitted for brevity):

$$\begin{aligned} \frac{\partial f}{\partial t} &= \frac{\alpha^2}{2}\frac{\partial^2 f}{\partial x^2} - \beta\frac{\partial f}{\partial x} \\ &\quad + \sum_{i=1}^{n} P_i(t) f^{lower}(x) \lambda_i^{lower}(1-a_i) \\ &\quad + \sum_{i=1}^{m} Q_i(t) f^{upper}(x) \lambda_i^{upper}(1-b_i) \end{aligned} \tag{3.9}$$

with the following initial and boundary conditions, where $\delta^*(x)$ is the Dirac delta function:

$$\begin{aligned} f(0,t|x_0) &= 0 & t \geq 0 \\ f(K,t|x_0) &= 0 & t \geq 0 \\ f(x,0|x_0) &= \delta^*(x-x_0) & 0 < x < K \end{aligned}$$

$$P_1(0) = \begin{cases} 0 & x_0 > 0 \\ 1 & x_0 = 0 \end{cases} \tag{3.10}$$

$$Q_1(0) = \begin{cases} 0 & x_0 < K \\ 1 & x_0 = K \end{cases}$$

Differential equations describing probability mass flow between the stages of the lower (and also upper) boundary are obtained based on the probability mass conservation equation. For the lower boundary we obtain:

$$\frac{dP_i(t)}{dt} = \begin{cases} -\lambda_1^{lower} P_1(t) + \left[\frac{\alpha^2}{2}\frac{\partial f}{\partial x} - \beta f\right]_{x=0} & i = 0 \\ -\lambda_i^{lower} P_i(t) + \lambda_{i-1}^{lower} a_{i-1} P_{i-1}(t) & 1 < i \leq n \end{cases} \tag{3.11}$$

Similarly, for the upper boundary we get:

$$\frac{dQ_i(t)}{dt} = \begin{cases} -\lambda_1^{upper} Q_1(t) + \left[\frac{\alpha^2}{2}\frac{\partial f}{\partial x} - \beta f\right]_{x=K} & i = 0 \\ -\lambda_i^{upper} Q_i(t) + \lambda_{i-1}^{upper} b_{i-1} Q_{i-1}(t) & 1 < i \leq m \end{cases} \tag{3.12}$$

3.3 Diffusion approximation for a single queue

We use the two-barrier diffusion equation introduced in Section 3.2 as an approximation of the queuing process $N(t)$. Consider single communication link as shown on Figure 3.5. 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.
- K: queue capacity.

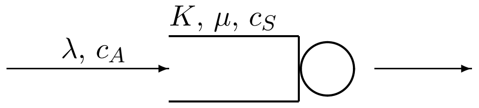


Figure 3.5: Diffusion approximation of single server queue.

The time evolution of $f(x, t|x_0)$ is the solution to the Equation 3.9 with initial and boundary conditions 3.10. The time evolution of the probability $P_i(t)$ of being at time t at the i-th stage of the lower boundary is the solution to Equation 3.11 also with initial and boundary conditions 3.10. Similarly, the time evolution of the probability $Q_i(t)$ of being at time t at the i-th stage of the upper boundary is the solution to Equation 3.12 also with initial and boundary conditions 3.10. Parameters β and α^2 are set to $\lambda - \mu$ and $\lambda c_A + \mu c_S$, respectively. The return functions $f^{lower}(x), x \in (0, K)$ and $f^{upper}(x), x \in (0, K)$ reflect the fact that the particle after being held at the upper boundary returns to state $K-1$ and after being held at the lower boundary to state 1. Thus we substitute:

$$\begin{aligned} f^{lower}(x) &= \delta^*(x-1) \\ f^{upper}(x) &= \delta^*(x-K+1) \end{aligned} \tag{3.13}$$

where $\delta^*(x)$ is the Dirac delta function.

The Laplace transform $f^*(x,s|x_0)$ of $f(x,t|x_0)$ can be computed in terms of the Laplace transform of the lower holding time distribution, denoted by $h^*(s)$ and upper holding time distribution $H^*(s)$. The solution is as follows [28]:

$$\begin{aligned} f^*(x,s|x_0) &= v^*(x,s|x_0) + v^*(x,s|1) \\ &* \; e^{-\frac{\beta}{\alpha^2}x_0} \frac{[\cosh(Ax_0) - Z_1 \sinh(Ax_0)]\, h^*(s)}{1 - h^*(s) e^{-\frac{\beta}{\alpha^2}} \; [\cosh(A) - Z_1 \sinh(a)]} \\ &+ \; v^*(x,s|K-1) e^{-\frac{\beta}{\alpha^2}(K-x_0)} \\ &* \; \frac{\{\cosh[A(K-x_0)] - Z_2 \sinh[A(K-x_0)]\}\, H^*(s)}{1 - H^*(s) e^{\frac{\beta}{\alpha^2}} [\cosh(A) - Z_2 \sinh(A)]} \end{aligned} \tag{3.14}$$

where

$$\begin{aligned} v^*(x,s|x_0) &= \frac{e^{\frac{\beta}{\alpha^2}(x-x_0)}}{\alpha^2 A} \left[e^{-A|x-x_0|} - e^{-A|x+x_0|} - 2e^{-2AK} \frac{\sinh(Ax_0)}{\sinh(AK)} \sinh(Ax) \right] \\ Z_1 &= \frac{\cosh(AK) - H^*(s) e^{\frac{\beta}{\alpha^2}} \cosh[A(K-1)]}{\sinh(AK) - \; H^*(s) e^{\frac{\beta}{\alpha^2}} \sinh[A(K-1)]} \\ Z_2 &= \frac{\cosh(AK) - h^*(s) e^{-\frac{\beta}{\alpha^2}} \cosh[A(K-1)]}{\sinh(AK) - \; h^*(s) e^{-\frac{\beta}{\alpha^2}} \sinh[A(K-1)]} \\ A &= \frac{\sqrt{2\alpha^2 s + \beta^2}}{\alpha^2} \end{aligned} \tag{3.15}$$

Equations 3.11 and 3.12 have solutions:

$$\begin{aligned}
P_i^*(s) &= e^{-\frac{\beta}{\alpha^2}x_0}\frac{[\cosh(Ax_0) - Z_1\sinh(Ax_0)]\ \frac{a_i}{\lambda_i^{lower}}e_i^*(s)}{1 - h^*(s)e^{-\frac{\beta}{\alpha^2}}[\cosh(A) - Z_1\sinh(A)]} \\
Q_i^*(s) &= e^{-\frac{\beta}{\alpha^2}(K-x_0)}\frac{\{\cosh A[(K - x_0)] - Z_2\sinh[A(K - x_0)]\}\ \frac{b_i}{\lambda_i^{upper}}E_i^*(s)}{1 - H^*(s)e^{\frac{\beta}{\alpha^2}}[\cosh(A) - Z_1\sinh(A)]}
\end{aligned} \tag{3.16}$$

where

$$\begin{aligned}
e_i^*(s) &= \Pi_{j=1}^{i}\frac{\lambda_j^{lower}}{\lambda_j^{lower} + s} \quad , i = 1, \ldots, n. \\
E_i^*(s) &= \Pi_{j=1}^{i}\frac{\lambda_j^{upper}}{\lambda_j^{upper} + s} \quad , i = 1, \ldots, m.
\end{aligned} \tag{3.17}$$

The Laplace transform $P^*(t) = \sum_{i=1}^{n} P_i^*(t)$ and $Q^*(t) = \sum_{i=1}^{n} Q_i^*(t)$ of the probabilities of being at the boundaries are given by:

$$\begin{aligned}
P^*(s) &= \frac{1 - h^*(s)}{s}e^{-\frac{\beta}{\alpha^2}x_0}\frac{[\cosh(Ax_0) - Z_1\sinh(Ax_0)]}{1 - h^*(s)e^{-\frac{\beta}{\alpha^2}}[\cosh(A) - Z_1\sinh(A)]} \\
Q^*(s) &= \frac{1 - H^*(s)}{s}e^{-\frac{\beta}{\alpha^2}(K-x_0)}\frac{\{\cosh[A(K - x_0)] - Z_2\sinh[A(K - x_0)]\}}{1 - H^*(s)e^{\frac{\beta}{\alpha^2}}[\cosh(A) - Z_1\sinh(A)]}
\end{aligned} \tag{3.18}$$

Note that $f^*(x, s|x_0)$, $P^*(t)$ and $Q^*(t)$ depend only on the Laplace transforms of lower and upper holding times and not explicitly on the parameters of the Coxian distributions. Since Coxian distribution is almost general (i.e., can fit any number of moments of an arbitrary distribution), we can use Laplace transforms of distributions of choice for $h^*(s)$ and $H^*(s)$.

3.4 Numerical inversion of the Laplace transform

There are several methods to numerically invert the transform. We use Stehfest's algorithm [41]. Assume that the Laplace transform is given as a function $f^*(s)$ for a complex number s. We obtain the value of time-domain function $f(t)$ at instant t according to:

$$\begin{aligned} f(t) &= \frac{\ln 2}{t} \sum_{i=1}^{L} V_i f^* \left(\frac{\ln 2}{t} i \right) \\ V_i &= (-1)^{\frac{L}{2}+i} \sum_{k=\frac{i+1}{2}}^{Min(i,\frac{L}{2})} \frac{k^{\frac{L}{2}+1}(2k)!}{(\frac{L}{2}-k)!k!(k-1)!(i-k)!(2k-i)!} \end{aligned} \tag{3.19}$$

The constant L should be chosen based on the desired accuracy and available precision of floating-point operations. We use $L = 16$ which results in a precision of four significant digits. Note that values V_i in Equation 3.19 depend only on K and may be precomputed.

3.5 Quality of diffusion approximation

Using the first two moments of arrival and service processes to obtain diffusion approximations of steady-state queue size for GI/G/1 queues has been extensively studied in the literature. In case of simpler queuing systems, it is possible to compare closed-form solutions obtained using diffusion approximation with known analytical formulas. In particular, in case of M/G/1 queuing system with mean arrival rate λ, mean service rate μ (with $\rho = \frac{\lambda}{\mu}$), and squared coefficient of variation of service time c_S, the steady-state mean queue size $\bar{N}_{PK}$ is given by Pollaczek-Khintchine formula [4]:

$$\bar{N}_{PK} \quad = \quad \rho \left[1 + \frac{\rho(1 + c_S)}{2(1 - \rho)} \right] \tag{3.20}$$

Steady-state solution of Equation 3.9 may be obtained as:

$$\bar{N}_{Diff} \quad = \quad \rho \left[\frac{1}{2} + \frac{\rho c_A + c_S}{2(1 - \rho)} \right] \tag{3.21}$$

Hence, the absolute error of the diffusion approximation in this case is:

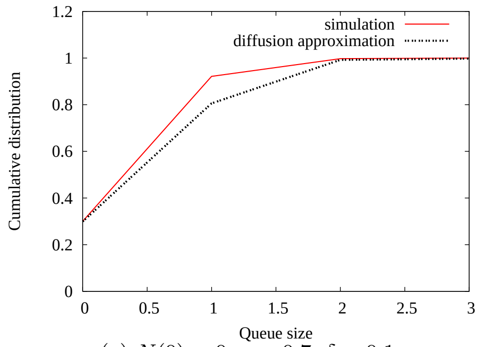
$$N_{PK} - N_{Diff} \quad = \quad \frac{\rho}{2} (1 - c_S) \tag{3.22}$$

Note that the relative error approaches 0 with ρ approaching 1 (heavy traffic situation). For a general GI/G/1 queuing system, no exact closed-form solution is known even for mean queue size in steady-state. However, studies reported in [17, 50, 51, 52] lead to the conclusion that the relative error of steady-state queue size approximation is on the order of $0.05c_A$.

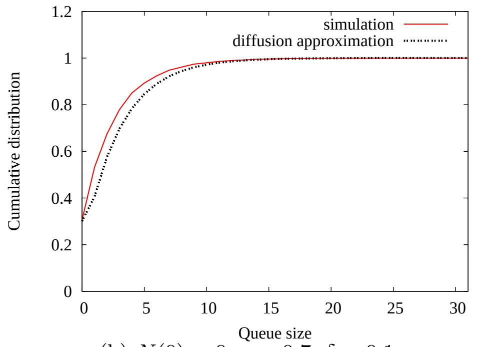
3.5.1 Transient queue size distribution

We are more interested in the quality of approximation for transient probability distribution, specifically $Pr[N(t + \delta)|N(t)]$, for which exact closed-form solution is not available. Also, the solution of Equation 3.9 may be obtained only in transformed domain. Thus we use simulations to quantify the quality of the approximation offered by the transient solution to the diffusion equation. Simulations cover configurations with the following values: ρ ranging from 0.1 to 1 (in steps of 0.05); c_A and c_S ranging from 0.1 to 2.0 (in steps of 0.2), initial queue size $N(t)$ ranging from 0 to 5, and 3 timestep sizes of 0.05, 0.1, and 0.15 The distributions used for generating service and arrival statistics were hyperexponential, hypoexponential, exponential, Pareto, and uniform. For each configuration, we compared the empirical cumulative distribution of queue size obtained using ns-2 [11] simulator against the transient predicted by the diffusion approximation. We found the average relative approximation error to be 3% with maximum error of 39%.

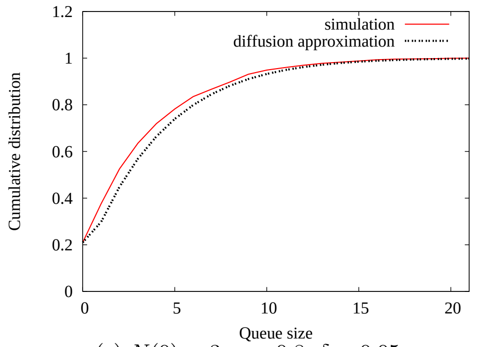
Figure 3.6 presents eight example comparisons of cumulative queue size distributions. Figures 3.6a and 3.6b present results for the same value of $\rho = 0.7$ and $\delta = 0.1$, but two different variability levels, namely $c_A = c_S = 0.1$ for (a) and $c_A = 0.7$, $c_S = 1.3$ for (b). Note, that although $N(0) = 0$ in both cases, higher variability of case (b) results in much larger range of likely states at time δ. Similar effect is depicted on plots 3.6c and 3.6d, where in both cases $\rho = 0.8$, $c_A = 1.1$, $c_S = 0.9$, and $N(0) = 3$, but the size of the timestep is different. Configuration 3.6c has $\delta = 0.05$, while 3.6d has $\delta = 0.15$. The resulting difference between queue size distributions at δ is significant. Figure 3.6e presents results for the configuration with low utilization and variability, specifically, $\rho = 0.15$, $c_A = c_S = 0.1$, and $\delta = 0.15$. Probability of $N(\delta) > 1$ is nearly 0. Finally, the opposite case is presented on plot 3.6f. Here both the variability and step size are large ($\rho = 0.9$, $c_A = 1.5$, $c_S = 1.7$, and $\delta = 0.15$) resulting in huge spread of likely states at δ. In all cases we observe excellent accuracy of diffusion approximation.



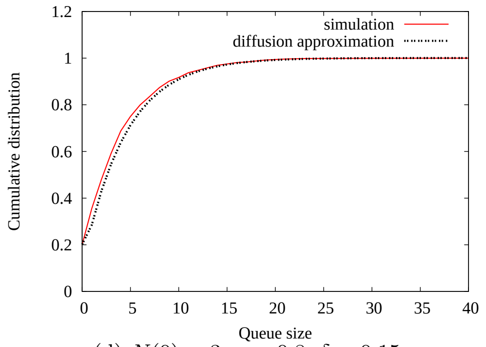
(a) $N(0) = 0,\ \rho = 0.7,\ \delta = 0.1$
$c_A = c_S = 0.1$



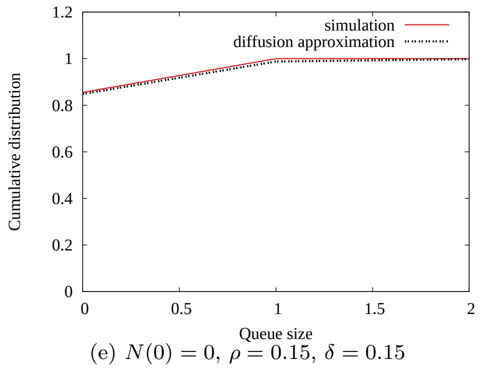
(b) $N(0) = 0,\ \rho = 0.7,\ \delta = 0.1$
$c_A = 0.7,\ c_S = 1.3$



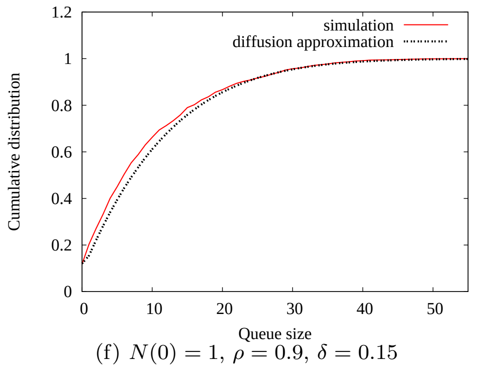
(c) $N(0) = 3,\ \rho = 0.8,\ \delta = 0.05$
$c_A = 1.1,\ c_S = 0.9$



(d) $N(0) = 3,\ \rho = 0.8,\ \delta = 0.15$
$c_A = 1.1,\ c_S = 0.9$



(e) $N(0) = 0,\ \rho = 0.15,\ \delta = 0.15$
$c_A = 0.1,\ c_S = 0.1$



(f) $N(0) = 1,\ \rho = 0.9,\ \delta = 0.15$
$c_A = 1.5,\ c_S = 1.7$

Figure 3.6: Comparison of $Pr[N(\delta) \leq x | N(0)]$ obtained by diffusion approximation and by packet-level simulation for 8 example configurations.

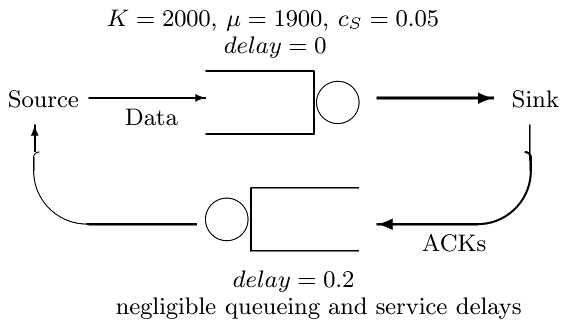

$K = 2000$, $\mu = 1900$, $c_S = 0.05$
$delay = 0$
Source
Data
Sink
ACKs
$delay = 0.2$
negligible queueing and service delays

Figure 3.7: Simple network with state-dependent traffic source.

The biggest errors are usually for estimation of $P[N(\delta) = 1|N(0)]$.

3.5.2 State-dependent UDP example

We now illustrate how TSS may be applied to solve for metrics of interest for a simple state-dependent traffic source operating over a single communication link. Consider the configuration shown in Figure 3.7. A state-dependent UDP source sends traffic to a sink via a communication link with buffer size $K = 2000$ packets, zero propagation delay, and service times uniformly distributed with mean $\mu = 1900$ packets/second and squared coefficient of variation of 0.05. The source generates fixed-size packets with exponentially distributed inter-arrival times. The sink sends an ACK for each data packet via a return link that has 0.2 second propagation delay and negligible queuing and service delays. (The coefficient of variation of a source corresponds to how much jitter the source exhibits with respect to its mean rate.)

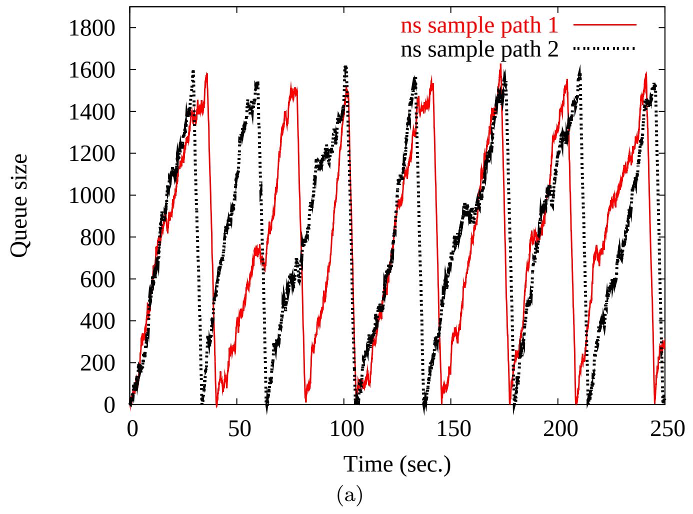
The source uses the incoming stream of acknowledgments to measure the roundtrip time (rtt) on the path, and uses this rtt to control its data rate. Whenever the measured rtt becomes smaller than 0.25 seconds, the source increases its rate to 1950 packets/second. Whenever the measured rtt becomes larger than

1.0 second, the source decreases its rate to 1500 packets/second.

Figures 3.8a and 3.8b show several sample paths generated using ns simulator and our TSS simulator (with $\delta = 0.05$ seconds). Clearly, both techniques generate very similar sample paths. TSS computes a sample path about *15 times* faster than ns, for the current link speed of 1900 packets/second, which, assuming 1KB packets, corresponds to a bandwidth of about 15 Mbps. If the bandwidth is increased to 1Gbps, TSS is faster by a factor of 1500, because the time required by TSS is independent of the link bandwidth whereas the time required by a packet-level simulator (such as ns) increases dramatically with link speed. Packet-level simulator actually scales worse than linearly because of non-linear effects such as caching and virtual memory.

Although both TSS and ns generate very similar sample paths, we do not expect two sample paths (even if generated by the same simulator) to be the same (unless the same random number stream was used in both). But we can compare the time evolutions of ensemble metrics produced by the two techniques. Figure 3.9a depicts the time evolutions of $E[N(t)]$, the instantaneous ensemble-average queue size as computed by ns and by TSS. As we see, the two are practically identical, confirming the high accuracy of the TSS approach. This figure also shows that TSS captures the time decay in the amplitude of the cyclic evolution of $E[N(t)]$, caused by the random arrival and service times and state-dependent control. This effect would not be captured by a technique, such as fluid approximation, that computes only one evolution of the system; there, the evolution would be cyclical but would not decay over time.

Another advantage of TSS is that it may be used to compute the distributions of metrics just as in discrete-event simulation. Figure 3.9b shows the distribution



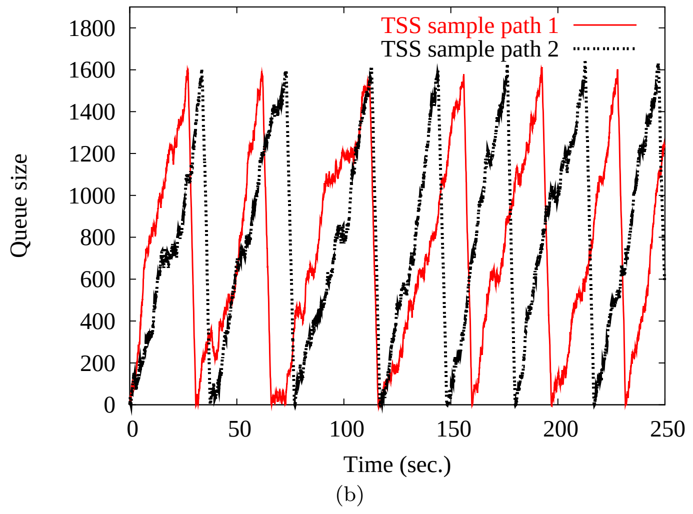
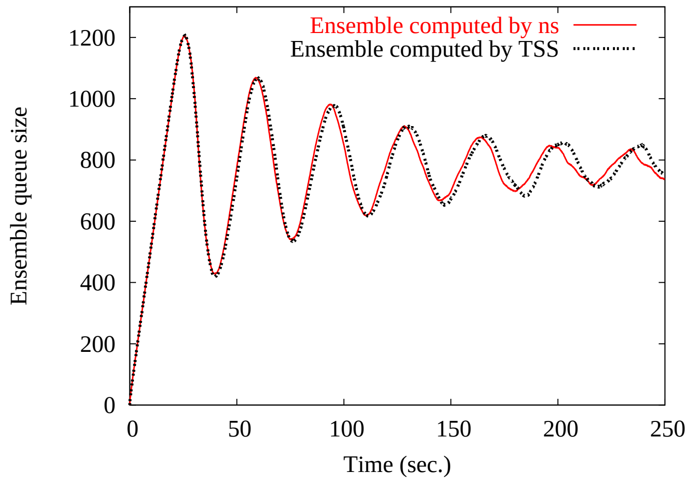
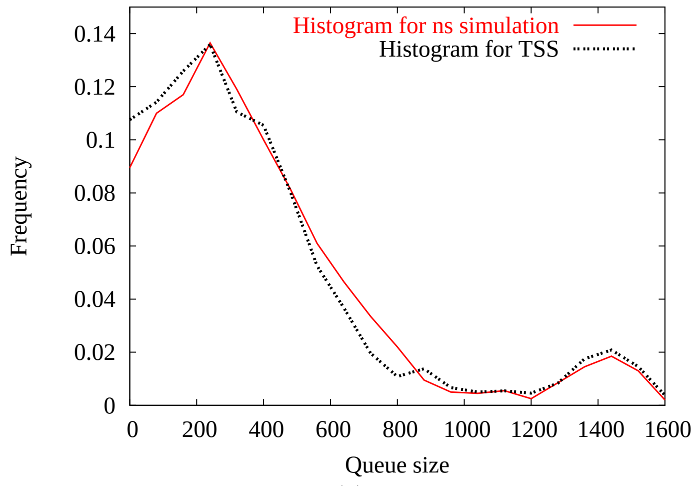


Figure 3.8: For the source-to-sink link in example network from Figure 3.7 two ns-generated sample paths (a), and two TSS-generated sample paths (b) of queue size $N(t)$.

of the queue size $N(t)$ for $t = 40$ seconds as computed by ns and by TSS. Again, the match is excellent. Such a metric cannot be obtained by a fluid approximation, which does not yield distributions.



(a)



(b)

Figure 3.9: For the source-to-sink link in example network in Figure 3.7, time evolutions of the instantaneous ensemble-average queue size $E[N(t)]$ (a), and histogram of the queue size $N(t)$ at $t = 40$ seconds (b) as computed by TSS (dashed line) and by ns (solid line). Each is averaged over 2000 runs.

Chapter 4

Extension to Networks of Queues

The previous chapter described Timestep Stochastic Simulation of a single link. To extend this 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. As usual, this assumption is valid because we set δ to be smaller than the control time scale Δ.

4.1 Departure process

We start with the problem of obtaining the first two moments of the departure process of a queue. Obtaining the mean is, of course, easy. Obtaining the second moment is very difficult, because the departure process of a queue is, in general, not a renewal process (except for a few simple cases such as M/M/1 queue in steady-state). We follow the approach of Whitt [16], Duda [27], Kuehn [26] and others by approximating the departure process by a renewal process with matching first two moments. The starting point here is the fundamental relationship derived by Marshall [38] between the Laplace transforms of service time distribution $g(s)$, idle period distribution $h(s)$, and inter-departure time distribution $\phi(s)$, all at steady-state:

$$\phi(s) = (1 - \pi_0)g(s) + \pi_0 g(s)h(s) \tag{4.1}$$

where π_0 is the probability that arriving customer finds the system empty.

This relationship may be used to approximate the first two moments of the departure process ([27],[28],[16]) for various queuing systems in steady-state. We need a time-dependent version of Equation 4.1. Assume that the only time-dependent element in the formula is the probability π_0 of an arrival finding an empty queue. Then we can approximate the transient departure process by using Marshall's formula with π_0 replaced with $\pi_0(t)$ as computed by the diffusion approximation. Differentiation of Equation 4.1 leads to:

$$\begin{aligned} E[\phi(t)] &= \frac{1}{\mu} + \pi_0(t)E[h] \\ E[\phi(t)^2] &= V_s + \frac{1}{\mu^2} + \pi_0(t)\left\{E[h^2] + 2\frac{E[h]}{\mu}\right\} \end{aligned} \tag{4.2}$$

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.

Figure 4.1 shows the first two moments of the departure process of a queue averaged over an interval of size $\delta = 0.05$ seconds, as computed by ns and by TSS. The x-axis is the queue size $N(t)$ at the beginning of the interval. The y-axis represents the mean and the standard deviation of the inter-departure time averaged over the interval. The results, obtained by averaging over 1000

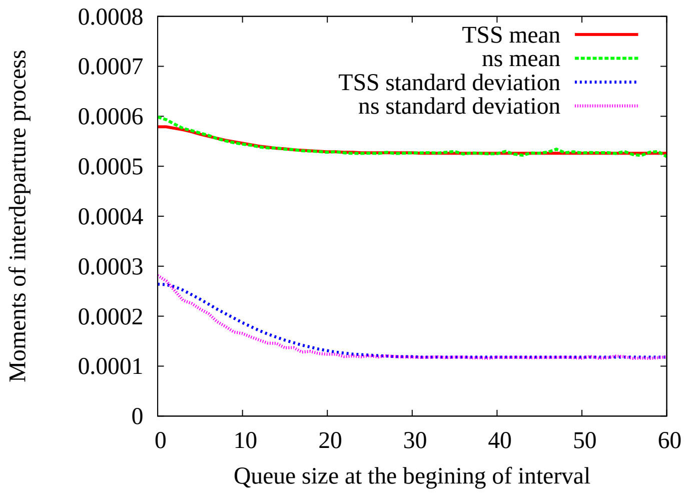


Figure 4.1: Comparison of the first two moments of queue departure process over interval $[t, t+\delta]$ vs $N(t)$ as computed by ns and TSS for $\lambda = 1800$, $\mu = 1900$, $K = 300$, $c_A = 1$, $c_S = 0.05$.

15-second runs, show excellent agreement between TSS and the simulation. The queue has exponential arrivals of mean rate 1800, maximum queue size $K = 300$, and uniformly distributed service times with mean $1/1900$ and squared coefficient of variation 0.05.

4.2 Merging processes

Consider the merge of n independent renewal processes (Figure 4.2). Analytical solution of the distribution of the merged process is known only for the simplest case when all component processes are Poisson. In such situation the merged process is also Poisson with rate equal to the sum of the arrival rates of the component processes. However, if at least one of the component processes is not Poisson, the merged process is not even a renewal process. We use an approximating renewal process to capture the statistics of the merged process.

There are several approximation methods that may be used to determine

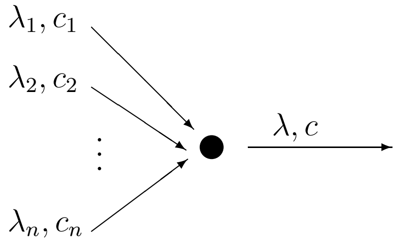


Figure 4.2: Merging n component processes.

the first two moments of the merged process [53, 26, 28, 27, 19]. We use a combination of two of these methods, namely the *asymptotic method* and the *stationary-interval method*. The first approaches the correct solution as the traffic intensity of the link (to which the merged traffic is fed) approaches 1, whereas the second is asymptotically correct as n approaches infinity. By appropriately combining the two, a value can be obtained that works for an adequate range.

4.2.1 Asymptotic method

The asymptotic method can be used to merge n component processes. For $i = 1, 2, \ldots, n$ assume that the mean arrival rate and squared coefficient of variation of process i are λ_i and c_i, respectively. Then mean arrival rate λ and squared coefficient of variation c of the merged process are given by:

$$\begin{aligned} \lambda &= \sum_{i=1}^{n} \lambda_i \\ c &= \sum_{i=1}^{n} \frac{\lambda_i}{\lambda} c_i \end{aligned} \tag{4.3}$$

4.2.2 Stationary interval method

The stationary interval method may be used for merging two renewal processes; to merge n processes, the method is applied $n-1$ times for subsequent pairs of processes until we are left with one process. Consider the merge of two independent renewal processes as shown in Figure 4.3 having mean arrival rates and squared coefficients of variations λ_1, λ_2, and c_1, c_2, respectively.

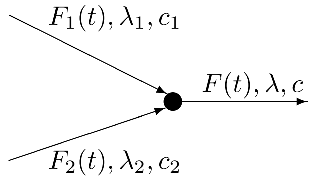


Figure 4.3: Merging 2 component processes.

It turns out that the first two moments of the merged process depend not just on the first two moments of the constituent renewal processes, but also on their distributions. Let $F_1(t)$ and $F_2(t)$ be their distributions. Following [26, 19, 53], the distribution of the inter-arrival time of the merged process $F(t)$ can be obtained from the distributions of the inter-arrival times of the composite processes, $F_1(t)$ and $F_2(t)$ as follows:

$$\begin{aligned} F(t) &= 1 - \frac{\lambda_1\lambda_2}{\lambda_1+\lambda_2}\Big\{F_1^c(t)\int_t^\infty F_2^c(u)du \\ &\qquad + F_2^c(t)\int_t^\infty F_1^c(u)du\Big\} \end{aligned} \tag{4.4}$$

where $F_i^c(t) = 1 - F_i(t)$ for $i = 1, 2$.

The first moment of the merged process is $\lambda = \lambda_1 + \lambda_2$. To obtain the second moment of the merged process, the so called substitution method can be used. We substitute the distribution $F_i(t)$ of each of the component processes by either a shifted exponential or a hyper-exponential distribution. The former is used if the squared coefficient of variation c_i is less than or equal than 1, and the latter if c_i is greater than 1. The parameters of the substitute process are determined based on the first two moments of the component process.

In particular, if $F_i(t)$ has mean arrival rate λ_i and squared coefficient of variation $c_i \geq 1$, $F_i(t)$ is replaced with a hyperexponential distribution of balanced means denoted *HyperExp(*a_1*,* a_2*,* p_1*,* p_2*)*, having density:

$$f(t) \quad = \quad p_1 a_1 e^{-a_1 t} + p_2 a_2 e^{-a_2 t} \quad , t \geq 0 \tag{4.5}$$

where:

$$\begin{aligned} p_1 &= \frac{1 + \sqrt{(c_i - 1)/(c_i + 1)}}{2} \\ p_2 &= 1 - p_1 \\ a_1 &= 2 p_1 \lambda_i \\ a_2 &= 2 p_2 \lambda_i \end{aligned} \tag{4.6}$$

If $c_i < 1$, $F_i(t)$ is replaced with an exponential distribution of mean rate h shifted by a constant d, denoted *ShiftExp(h, d)*, with density:

$$f(t) \quad = \quad h e^{-h(t-d)} \quad , t \geq d \tag{4.7}$$

where:

$$\begin{aligned} h &= \frac{\lambda_i}{\sqrt{c_i}} \\ d &= \frac{1}{\lambda_i} - \frac{1}{h} \end{aligned} \tag{4.8}$$

The squared coefficient of variation of the merged process is then determined. We first obtain the second moment of the interarrival time, I^2, as follows:

- For $F_1 = HyperExp(a_1, a_2, p_1, p_2)$ and $F_2 = HyperExp(b_1, b_2, q_1, q_2)$

$$I^2 = \frac{2\lambda_1\lambda_2}{\lambda_1 + \lambda_2} \sum_{i=1}^{2}\sum_{j=1}^{2} \frac{p_i q_j}{a_i b_j (a_i + b_j)} \tag{4.9}$$

- For $F_1 = ShiftExp(h_1, d_1)$ and $F_2 = ShiftExp(h_2, d_2)$

$$\begin{aligned} I^2 = \frac{2\lambda_1\lambda_2}{\lambda_1 + \lambda_2} &\left[\frac{d_1}{\lambda_1\lambda_2} - d_1^2 \frac{\lambda_1 + \lambda_2}{2\lambda_1\lambda_2} + \frac{d_1^3}{3} + \frac{1 - e^{-h_1(d_2 - d_1)}}{\lambda_2 h_1^2} \right. \\ &\left. + \frac{(1 + h_1 d_2) e^{-h_1(d_2 - d_1)} - (1 + h_1 d_1)}{h_1^3} + \frac{e^{-h_1(d_2 - d_1)}}{h_1 h_2 (h_1 + h_2)}\right] \end{aligned} \tag{4.10}$$

- For $F_1 = HyperExp(a_1, a_2, p_1, p_2)$ and $F_2 = ShiftExp(h, d)$

$$\begin{aligned} I^2 = &\frac{e^{a_1 d} p_1}{a_1 h (a_1 + h)} + \frac{e^{-a_2 d} p_2}{a_2 h (a_2 + h)} + \frac{p_1 (1 - e^{-a_1 d} - a_1 d e^{-a_1 d})}{a_1^3} \\ &- \frac{p_2 (1 - e^{-a_2 d} - a_2 d e^{-a_2 d})}{a_2^3} + \left(\frac{1}{h} + d\right) \left[\frac{p_1 (1 - e^{-a_1 d})}{a_1^2} \right. \\ &\left. + \frac{p_2 (1 - e^{-a_2 d})}{a_2^2}\right] \end{aligned} \tag{4.11}$$

Finally, we obtain the squared coefficient of variation of the merged process as:

$$c = I^2(\lambda_1 + \lambda_2)^2 - 1 \tag{4.12}$$

4.2.3 Combining asymptotic and stationary interval methods

As mentioned before, two approximations described above complement each other in terms of accuracy. Albin [19] performed a series of simulations in order to determine the best convex combination of the two approximations resulting in the smallest error in approximation of steady-state queue size of the GI/G/1 queue fed with the merged arrival process. She proposed to use a weighted average of both approximations with weighting function depending on the effective number of merged flows and utilization of the queue. The effective number of active flows is defined as:

$$n^* = \frac{1}{\sum_{i=1}^{n}(\frac{\lambda_i}{\lambda})^2}.$$

Utilization of the queue to which the traffic arrives is defined as $\rho = \dfrac{\lambda}{\mu}$. In such a setting, assuming that squared coefficient of variation of the merged flow computed using stationary interval approximation and asymptotic method are c_S and c_A, respectively, the resulting squared coefficient of variation c of the merged flow is equal to:

$$c = w(\rho, n^*)c_A + (1 - w(\rho, n^*))c_S \tag{4.13}$$

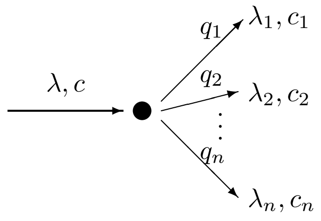


Figure 4.4: Splitting a flow in case of Bernoulli routing.

where

$$w(\rho, n^*) \quad = \quad \frac{1}{1 + 6(1-\rho)^{2.2} n^*} \tag{4.14}$$

This approach is very accurate in predicting mean steady-state queue size of GI/G/1 queue fed with merged traffic flow. Simulations reported in [19] show that average relative error is on the order of 3%.

4.3 Splitting a process

At a node of a network, the incoming traffic flow may be split into multiple flows, depending on the routing probabilities. We need to express the first two moments of the resulting flows based on the first two moments of the renewal process that is being split as well as on the routing probabilities. We assume that the routing probabilities remain constant within each δ-sized interval but may change between intervals.

Consider the configuration depicted in Figure 4.4. Let the inter-arrival time of the aggregate process being split have mean rate λ and squared coefficient of variation c. The packets may enter one of n communication links with probabilities $q_1, q_1, \ldots, q_n$ such that $\sum_i^n q_i = 1$. Let the inter-arrival time of the ith resulting process have mean rate λ_i and squared coefficient of variation c_i.

Following [4] and [26], the transition rates and squared coefficients of variation of the ith component process are given by:

$$\begin{aligned} \lambda_i &= \lambda q_i \\ c_i &= q_i c + (1 - q_i) \end{aligned} \tag{4.15}$$

4.4 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. Please refer to Table 4.1 for notation. 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:

$$A_i^j(t) = \begin{cases} \lambda_i^j(t) * \delta & \text{if } i \text{ is the first-hop of flow } j \\ D_k^j(t) & \text{otherwise, where } k \text{ is the previous link} \\ & \text{on the path of flow } j \end{cases} \tag{4.16}$$

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 numerical inversion of Equation 3.18.

$\lambda_i^j(t)$	arrival rate of packets of flow j to link i during $[t, t+\delta]$
$c_{A,i}(t)$	squared coefficient of variation of interarrival time to link i during $[t, t+\delta]$
$\mu_i(t)$	service rate of link i during $[t, t+\delta]$
$c_{S,i}(t)$	squared coefficient of variation of service time of link i during $[t, t+\delta]$
$N_i^j(t)$	number of packets of flow j in a queue of link i at time t
$A_i^j(t)$	number of packets of flow j arriving to the link i during $[t, t+\delta]$
$L_i^j(t)$	number of packets of flow j lost due to overflow of the link i during $[t, t+\delta]$
$D_i^j(t)$	number of packets of flow j departing the link i during $[t, t+\delta]$
$N_i(t)$	number of packets in a queue of link i at time t
$A_i(t)$	number of packets arriving to the queue of link i during $[t, t+\delta]$
$L_i(t)$	number of packets lost due to overflow of the link i during $[t, t+\delta]$
$D_i(t)$	number of packets departing the link i during $[t, t+\delta]$
$P_i^{empty}(t)$	probability that queue i is empty during $[t, t+\delta]$
$P_i^{full}(t)$	probability that queue i is full during $[t, t+\delta]$

Table 4.1: Notation.

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:

$$L_i(t) \;=\; A_i(t) P_i^{full}(t) \qquad (4.17)$$

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:

$$L_i^j(t) \;=\; L_i(t) \frac{A_i^j(t)}{A_i(t)} \qquad (4.18)$$

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 techiques presented in Sections 4.1, 4.2, and 4.3. 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 Chapter 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:

$$D_i^j(t) \;=\; N_i^j(t) - N_i^j(t+\delta) + A_i^j(t) - L_i^j(t) \qquad (4.19)$$

Chapter 5

Source Model in TSS

This chapter describes how to model a source for use in TSS. Because TSS evaluates a computer network in timesteps, a traffic source model must also operate on the same timescale. Users of packet-level simulators are familiar with discrete models, where actions of the source are modeled at the level of packet arrivals and transmissions. For example, a packet-level model of a time-dependent UDP source would provide the simulator with the time instants of its packet generations. Whereas a TSS model of the same traffic source should provide the simulator with the statistics of the traffic generated by source during each timestep, in particular, the first two moments of the inter-generation time distribution during each timestep.

The following sections present TSS models of three kinds of traffic sources: time-dependent UDP, state-dependent UDP, and TCP. A time-dependent UDP source injects packets into the network with rate and variation of interarrival times based only on time and not the state of the network. This can be an appropriate model for simple UDP sources such as streaming audio without dynamic rate control. The state-dependent UDP source adapts the arrival rate and variability to the state of the network, such as measured round-trip time.

A source model has a state of sufficient detail to obtain the rate and coefficient of variation of its flow, and the state at the next timestep. For example, the state of a TCP connection would include its congestion window and timer values. The model also defines how the state is to be updated at each timestep, perhaps based on some history of the network state.

We assume that the state of the source changes only at timestep boundaries, i.e., at times $t_0, t_0 + \delta, t_0 + 2\delta, \ldots$. We also assume that arrival process is i.i.d. within the interval, i.e., there is no correlation between interarrival times and all interarrival times are drawn from a distribution having mean $\frac{1}{\lambda}$ and squared coefficient of variation c.

5.1 Time-dependent UDP source

As a first example of traffic source model in TSS consider a time-dependent UDP source. This source varies the way in which data packets are sent into the network based solely on time and not on network conditions, i.e., supplies TSS with mean arrival rate and squared coefficient of variation of interarrival times for each timestep $[t, t + \delta]$.

5.2 State-dependent UDP source

The second example of traffic source is a simple state-dependent UDP. The source operates in two possible states differing in mean arrival rate of packets and squared coefficient of variation of interarrival times. The state of the source changes in response to network conditions. Precisely, it uses smoothed round-trip time (*srtt*) to measure the network congestion; *srtt* is an average of several recent

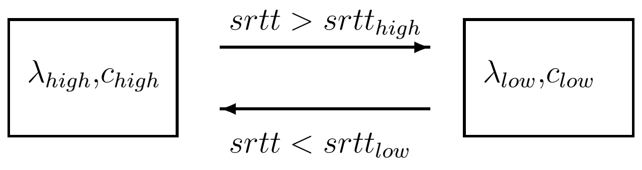


Figure 5.1: State-dependent UDP traffic source. Mean interarrival time and squared coefficient of variation changes depending on smoothed round-trip time.

round-trip time measurements. When *srtt* exceeds a fixed threshold $srtt_{high}$, the source switches to a lower sending rate λ_{low} and squared coefficient of variation c_{low}. When *srtt* falls below a certain threshold $srtt_{low}$, the source switches to higher sending rate λ_{high} and squared coefficient of variation c_{high}. The behavior of the source is depicted on Figure 5.1.

A packet-level representation of this source measures the round-trip time of each packet (using the stream of acknowledgments) and computes a running average of the last several measurements. Based on the computed average, the source adjusts its rate (according to Figure 5.1).

The TSS representation of this source performs the following actions for each step $[t, t+\delta]$:

- Compute *srtt* based on the queue sizes and propagation delays on all communication links on the flow's path.
- Set sending rate and coefficient of variation for the step based on the *srtt*.

5.3 TCP source

As a final example we present a timestep model of a TCP source. This illustrates the high level of detail that may be captured in TSS when simulating complex state-dependent congestion control schemes. Our model encapsulates TCP features such as slow-start, retransmissions, and congestion avoidance. This is possible because TSS models the sample path of the state of the TCP source (such as time evolution of congestion window).

We extend the information maintained at each queue for a TCP flow. In addition to the number of packets (Chapter 4) TSS also keeps track of the lowest and highest sequence number, denoted l_{seq} and h_{seq}, respectively. The number of packets of the flow at a queue is equal to $h_{seq} - l_{seq}$ if no packets have been lost; otherwise it is less than $h_{seq} - l_{seq}$. Counters h_{seq} and l_{seq} are updated at each timestep based on the arrival count and the loss count. The data arrival count and variability 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 state maintained by the source consists of the following pieces of information:

- **cwn** - congestion window size, i.e., maximum number of packets that may be "in-flight"
- **sst** - slow start threshold
- **lastSeq** - last acknowledged sequence number
- **rto** - rto timer value, i.e., time until which the last sent data segment has to be acknowledged or loss will be assumed

The source based on the statistics of arrival acknowledgment stream adjusts its congestion window. In particular, if the congestion window is smaller than the slow-start threshold it gets increased by the number of received acknowledgments.

Otherwise, if the source is in congestion avoidance mode, the congestion window is expanded by the number of arriving acknowledgments divided by the current size of the congestion window. Sending rate is determined based on the number of new acknowledgments received (i.e., acknowledgments with sequence numbers higher than last acknowledged segment).

Chapter 6

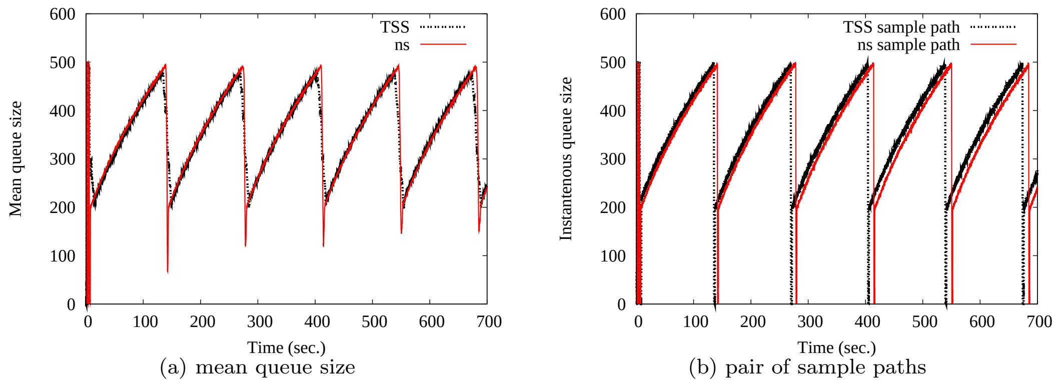
Simulation Studies

In this chapter we present a series of simulation studies comparing TSS to packet-level simulation for various network topologies and flow configurations. As a packet-level simulator, we used modified ns that allowed for random service times. While discussing the results we emphasize the accuracy of the TSS. Computational speed of TSS is 5 to 10 times higher than that of the packet-level simulation for the link speeds used here. For faster communication links, such as 2 Gbps, TSS is nearly 1000 times faster than ns. All TSS results reported in this chapter were obtained with timestep $\delta = 0.05$ second.

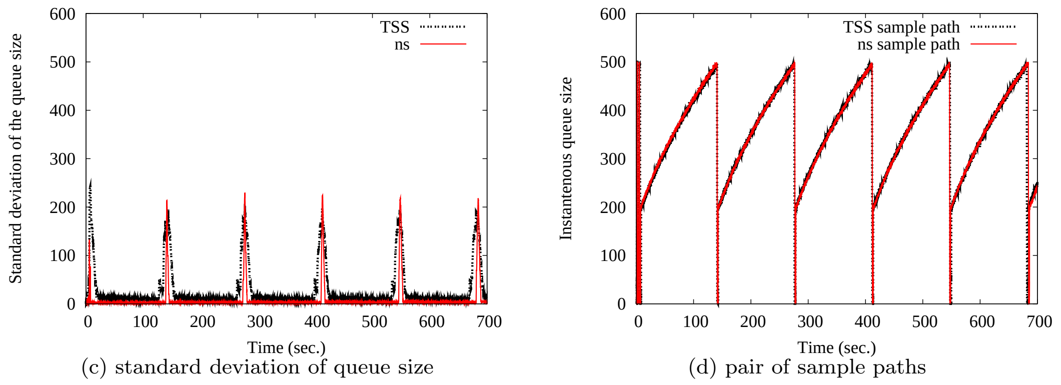
6.1 Single communication link

We start the evaluation of TSS with a simple network consisting of one communication link. Configuration is depicted on Figure 6.1. Data packets flow from node N_0 to N_1. Link from N_1 to N_0 is much faster and does not cause any queuing delay for acknowledgments. Service rate is 1000 packets/sec and we explore multiple coefficients of variation.

Figure 6.1: Single-link network.



(a) mean queue size
(b) pair of sample paths



(c) standard deviation of queue size
(d) pair of sample paths

Figure 6.2: Comparison of simulation results with TSS for link $N_0 \rightarrow N_1$ in Scenario 1 for network from Figure 6.1. Only one TCP flow is active throughout the simulation. Service rate of the link is 1000 packets/sec with squared coefficient of variation of 0.05.

Scenario 1

The service rate of the link is 1000 packets/sec with squared coefficient of variation of service time equal to 0.05. The buffering capacity of the link is 500 packets.

Figures 6.2a and 6.2c present time evolutions of mean queue size and its standard deviation, respectively. The results were obtained by averaging over

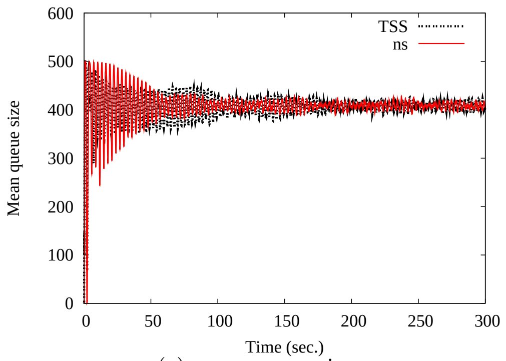
100 repetitions. Note the very high accuracy of the approximation for both mean queue size as well as the standard deviation. TSS captures properly the increased variability around points where queue overflows. Variability is due to the fact that the exact moment when the queue overflows is random. This is easily seen from the sample paths of the queuing process. Two pairs of such paths are depicted on Figures 6.2b and 6.2d.

Scenario 2

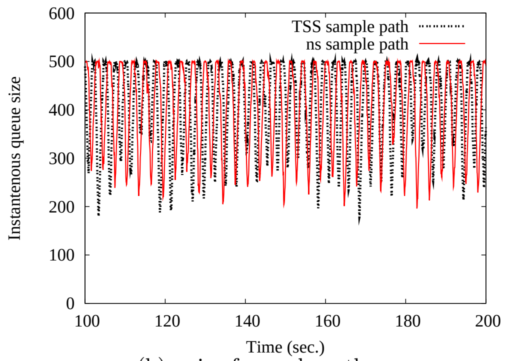
We now consider the same configuration as Scenario 1 except that instead of 1 TCP flow the link is shared by 100 TCP flows. In this case the frequency of queue fluctuations is of course much higher than with one flow. This can be seen on Figures 6.3b and 6.3d which contain two pairs of sample paths for this case. Another important observation is that since the queue overflows very often (compared to 1-flow case), the mean queue size as well as its standard deviation do not exhibit oscillatory behavior. Evolutions of mean queue size and its standard deviation are presented in Figures 6.3a and 6.3c. Note the very high accuracy with which TSS predicts both metrics of interest.

Scenario 3

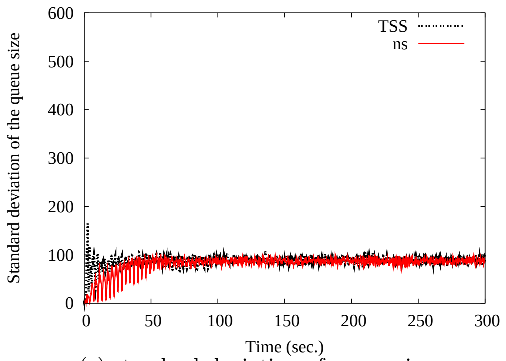
Consider the single link configuration with 1 TCP flow but much higher variability of service. Squared coefficient of variation in this case equals 1.2. The mean queue size looks similar to the first 1-flow example with low variability, however the sample paths, shown on Figures 6.4b and 6.4d are much more jittery. Both mean queue size as well as its standard deviation exhibit cyclic behavior as shown on Figures 6.4a and 6.4c.



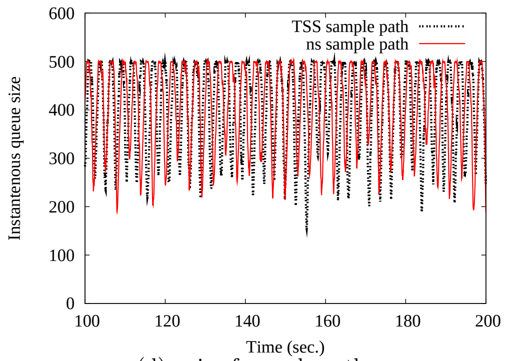
(a) mean queue size



(b) pair of sample paths

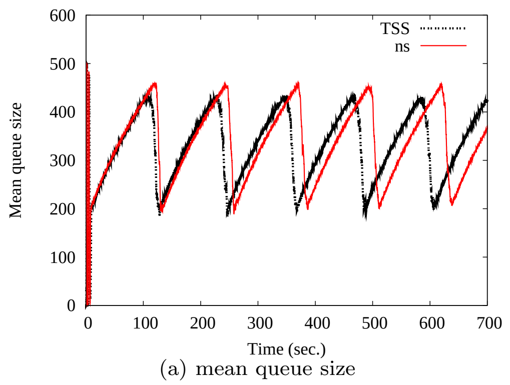


(c) standard deviation of queue size

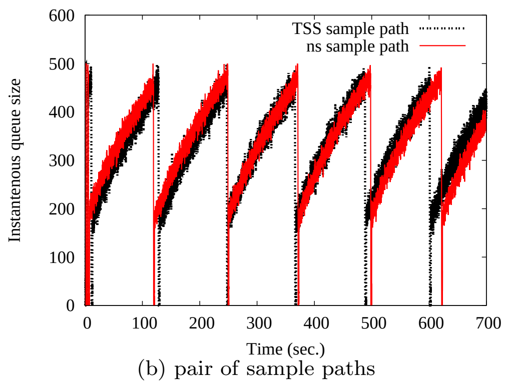


(d) pair of sample paths

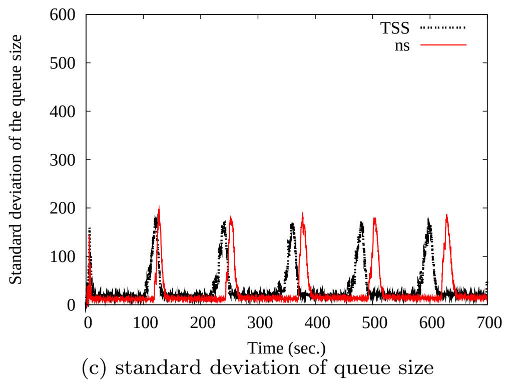
Figure 6.3: Comparison of simulation results with TSS for link $N_0 \longrightarrow N_1$ in Scenario 2 for network from Figure 6.1. The arrival traffic to the link is due to 100 TCP flows. Service rate of the link is 1000 packets/sec with squared coefficient of variation of 0.05.



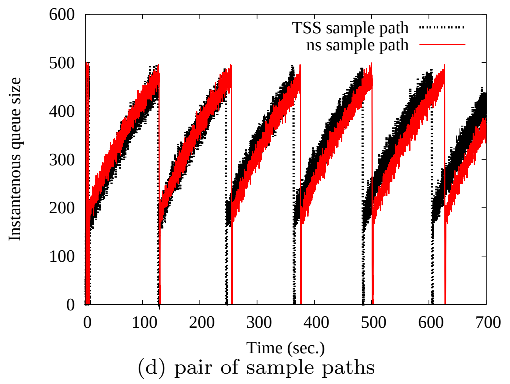
(a) mean queue size



(b) pair of sample paths



(c) standard deviation of queue size



(d) pair of sample paths

Figure 6.4: Comparison of simulation results with TSS for link $N_0 \longrightarrow N_1$ in Scenario 3 for network from Figure 6.1. Only one TCP flow is active throughout the simulation. Service rate of the link is 1000 packets/sec with squared coefficient of variation of 1.2.

Scenario 4

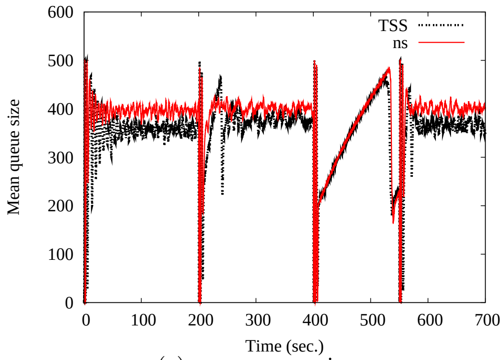
Finally, in the last example for this configuration the link is shared by varying number of TCP flows. 20 TCP flows are active from the beginning of the simulation to 200 second. At 200 second all 20 flows stop and another 4 flows start. At time 400 seconds all four flows stop and only one flow is started and remains active until 550 second. At 550 second 10 flows become active and continues transmission till the end of the simulation. Service rate of the link is 1000 packets/sec with squared coefficient of variation of 0.3. Figures 6.5a and 6.5c present time evolution of mean queue size and its standard deviation. Observe significant drop in variability of the queue size process between 400 and 550 second when only one flow is active as compared to 0 to 200 second when 20 flows are active. TSS predicts the behavior of the system very well for both the mean and the variability. Also two pairs of sample paths are presented on Figures 6.5b and 6.5d. TSS evolutions mimic those generated by packet-level simulation.

6.2 Two links in tandem

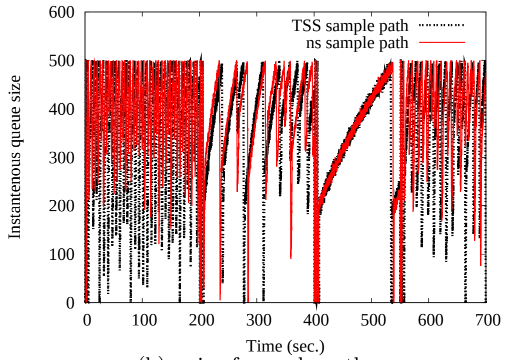
In this scenario we examine quality of TSS approximation for two communication links in tandem. Configuration is shown in Figure 6.6. Traffic flows from node N_0 to N_2.

Scenario 1

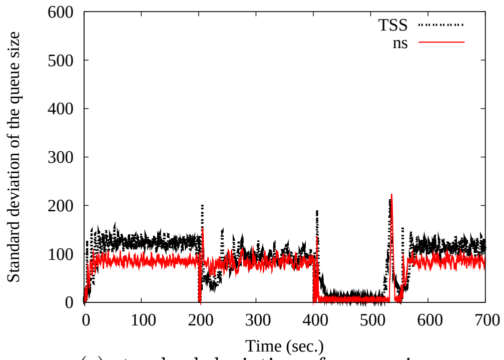
We examine the case where the two links are identical, specifically they have equal service rates of 1000 packets/sec, squared coefficient of variation of service time of 0.05, and buffer capacity of 500 packets. There is one TCP flow sending



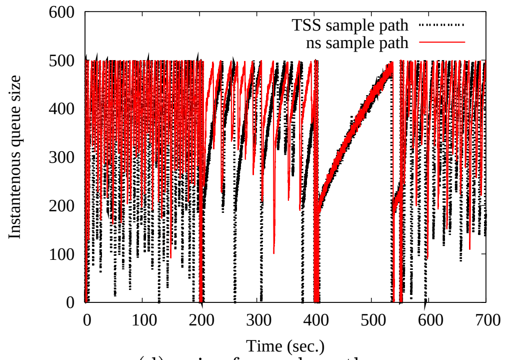
(a) mean queue size



(b) pair of sample paths



(c) standard deviation of queue size



(d) pair of sample paths

Figure 6.5: Comparison of simulation results with TSS for link $N_0 \rightarrow N_1$ in Scenario 4 for network from Figure 6.1. There is 35 TCP flows with time-dependent start times sharing the link. Service rate of the link is 1000 packets/sec with squared coefficient of variability of 0.3.

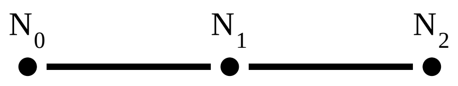


Figure 6.6: Two-link network.

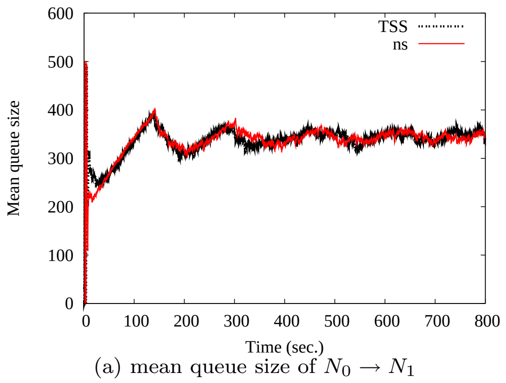
data from node N_0 to node N_2.

Figure 6.7 presents time evolutions of mean queue sizes and its standard deviation for both links for network configuration in Figure 6.6. Figures 6.7a and 6.7c show evolution of mean queue size for link $N_0 \rightarrow N_1$ and $N_1 \rightarrow N_2$, respectively. Figures 6.7b and 6.7d show corresponding pair of time evolutions of standard deviations of the queue size. TSS is very accurate in predicting the behavior of both links. The queuing on the second communication link ($N_1 \rightarrow N_2$) is due to the service rates being the same for both links. Since the first link is nearly always non-empty, the utilization of the second queue equals 1 for most of the time. This results in sample paths of wide diversity. Figures 6.8c and 6.8d show two pairs of sample paths as generated by packet-level simulation and TSS. Similar pairs for the first communication link are shown on Figures 6.8a and 6.8b.

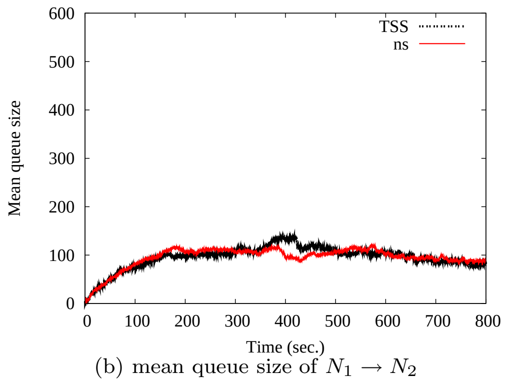
Scenario 2

Consider the tandem configuration with mean service rate of 2000 packets/sec for link $N_0 \rightarrow N_1$ and 1000 packets/sec for link $N_1 \rightarrow N_2$. Both links have squared coefficient of variation of service time equal to 0.05. Buffer capacities are 500 packets. There are 5 TCP flows sending data from N_0 to N_2.

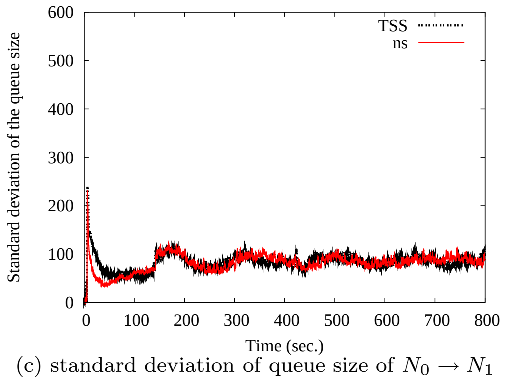
The second link ($N_1 \rightarrow N_2$) has high utilization and large queue size, whereas the first queue has low utilization. Comparison of the mean queue size and its standard deviation as computed by packet-level simulator and TSS for links $N_0 \rightarrow N_1$ are presented on Figures 6.9a and 6.9c. Figures 6.9b and 6.9d present corresponding results for link $N_1 \rightarrow N_2$. TSS has excellent accuracy not only for highly utilized queue but also for the one with low utilization. Corresponding pairs of sample paths as generated by packet-level simulation and TSS are shown



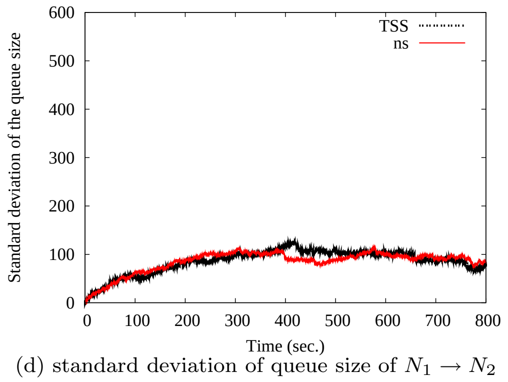
(a) mean queue size of $N_0 \rightarrow N_1$



(b) mean queue size of $N_1 \rightarrow N_2$

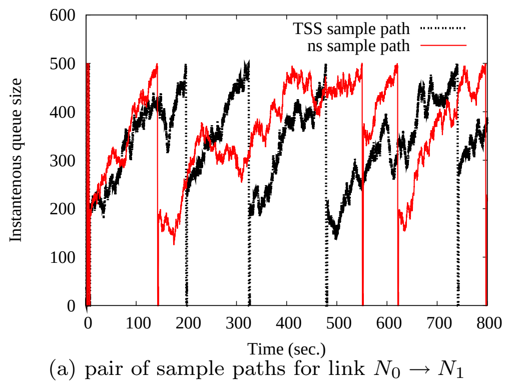


(c) standard deviation of queue size of $N_0 \rightarrow N_1$

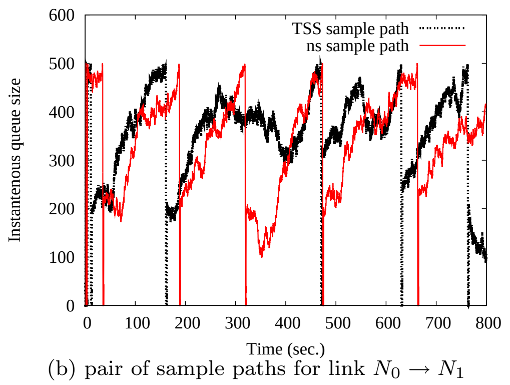


(d) standard deviation of queue size of $N_1 \rightarrow N_2$

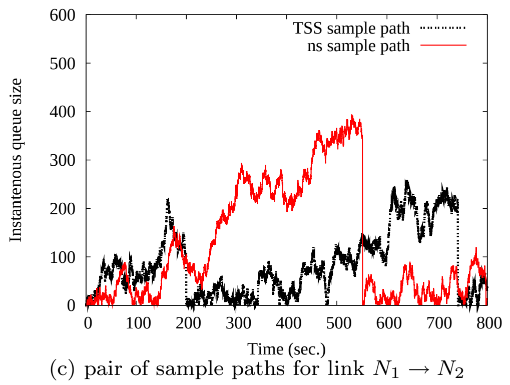
Figure 6.7: Comparison of simulation results with TSS for links $N_0 \longrightarrow N_1$ and $N_1 \longrightarrow N_2$ in Scenario 1 for network in Figure 6.6. Both communication links have service rate of 1000 packets/sec with squared coefficient of variation of service time equal to 0.05.



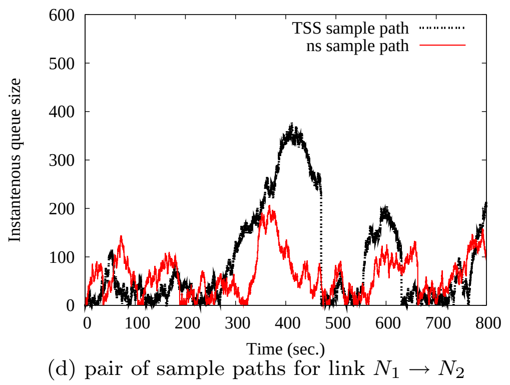
(a) pair of sample paths for link $N_0 \rightarrow N_1$



(b) pair of sample paths for link $N_0 \rightarrow N_1$

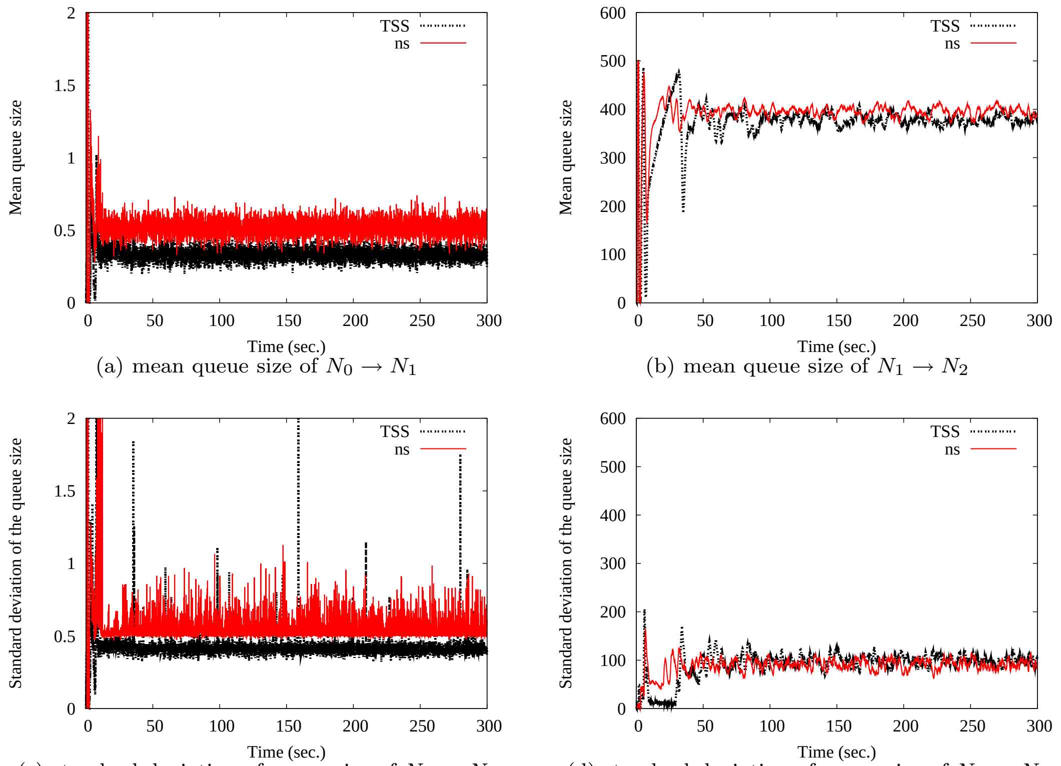


(c) pair of sample paths for link $N_1 \rightarrow N_2$



(d) pair of sample paths for link $N_1 \rightarrow N_2$

Figure 6.8: Comparison of sample paths generated by packet-level simulation with results obtained using TSS in Scenario 1 for tandem network in Figure 6.6.



(a) mean queue size of $N_0 \rightarrow N_1$ (b) mean queue size of $N_1 \rightarrow N_2$

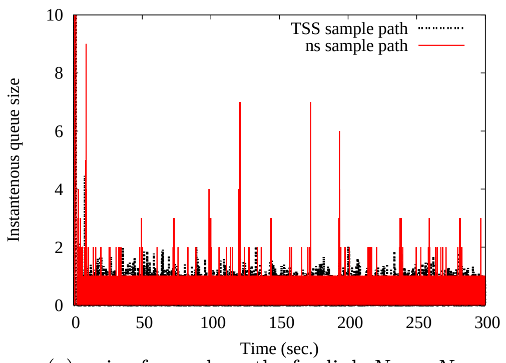
(c) standard deviation of queue size of $N_0 \rightarrow N_1$ (d) standard deviation of queue size of $N_1 \rightarrow N_2$

Figure 6.9: Comparison of simulation results with TSS for links $N_0 \longrightarrow N_1$ and $N_1 \longrightarrow N_2$ in Scenario 2 for network in Figure 6.6. Communication link $N_0 \longrightarrow N_1$ has service rate of 2000 packets/sec, $N_1 \longrightarrow N_2$ 1000 packets/sec. Both have squared coefficient of variation of service time equal to 0.05.

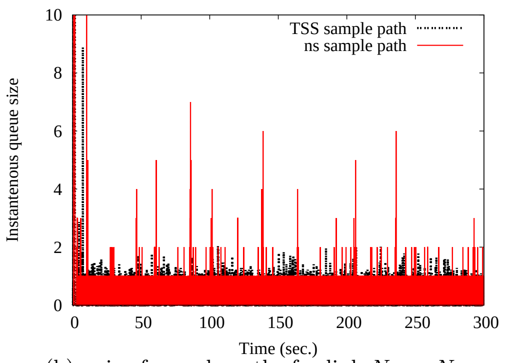
on Figure 6.10.

6.3 Six-node network

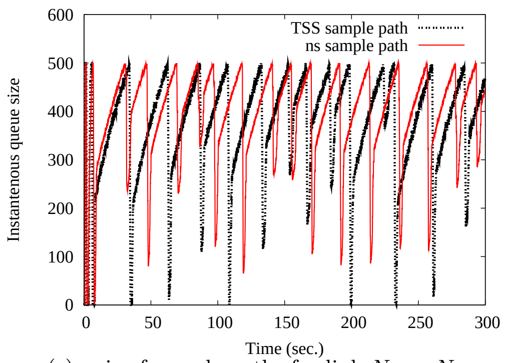
In this configuration we examine the approximation quality for six-node network presented in Figure 6.11. We evaluate the accuracy for two scenarios, one with bottleneck links on the edges of the network, and one with the bottleneck link being the central link ($N_2 \rightarrow N_3$).



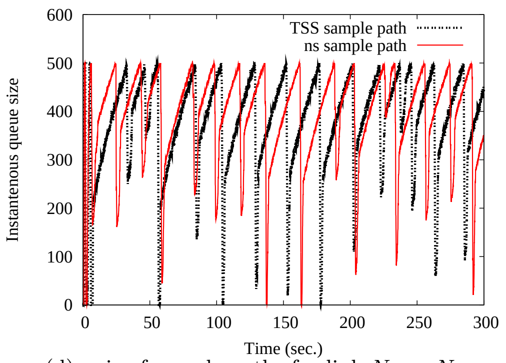
(a) pair of sample paths for link $N_0 \rightarrow N_1$



(b) pair of sample paths for link $N_0 \rightarrow N_1$



(c) pair of sample paths for link $N_1 \rightarrow N_2$



(d) pair of sample paths for link $N_1 \rightarrow N_2$

Figure 6.10: Comparison of sample paths generated by packet-level simulation with results obtained using TSS in Scenario 2 for the tandem network in Figure 6.6.

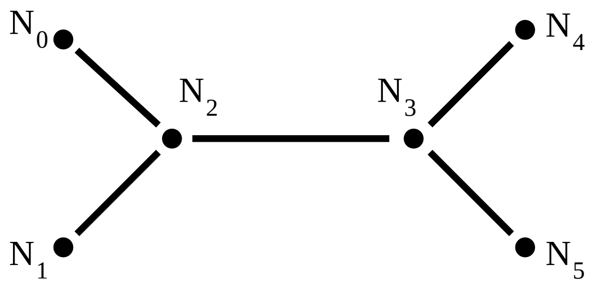
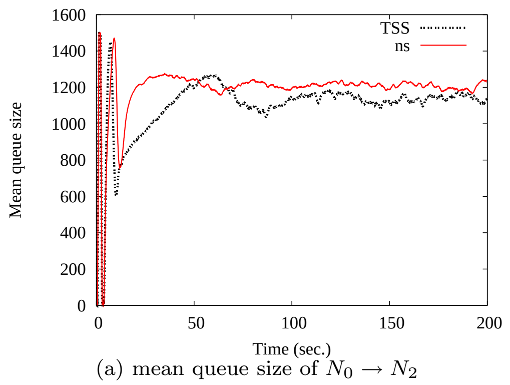


Figure 6.11: Six-node network.

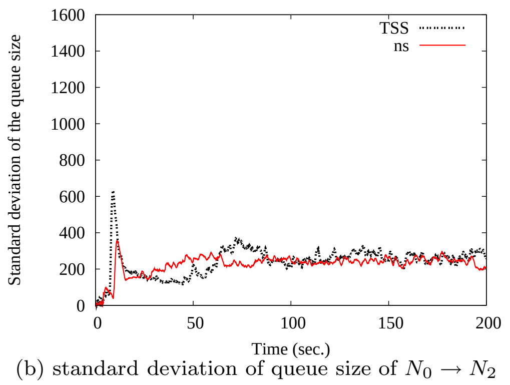
Scenario 1

Consider the configuration with service rate of 8000 packets/sec for the link $N_2 \rightarrow N_3$ and 2000 packets/sec for all remaining links. Squared coefficient of variation of service time for all links is 0.05. Buffering capacities of all links are 1500 packets. Traffic is generated by 10 TCP flows transferring data from nodes N_0 to N_4 and 10 TCP flows from nodes N_1 to N_5. Service rates of links used by acknowledgments are much higher thus there are no queuing delays for acknowledgments.

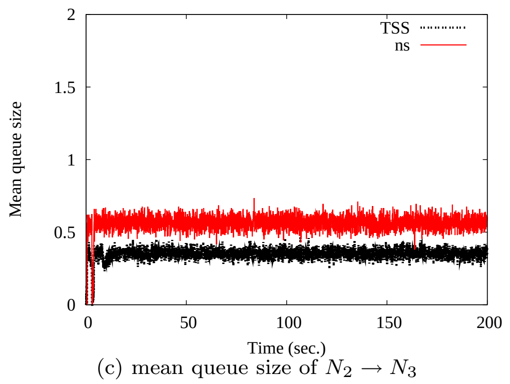
Figure 6.12 presents evolution of mean and standard deviation of queue size computed using 100 repetitions of packet-level simulation and TSS. Figures 6.12a and 6.12e present time evolution of mean queue size as computed by TSS and packet-level simulation for links $N_0 \rightarrow N_2$ and $N_3 \rightarrow N_4$, respectively.
Although service rate and its variability is the same for both links, the mean queue sizes are different. Link $N_0 \rightarrow N_2$ is the first hop on its TCP flows path and thus exhibits cyclic behavior. On the contrary, utilization of $N_3 \rightarrow N_4$ is nearly all the time 1, and its queue builds up only because of randomness in arrival and service processes. Note that TSS predicts time evolution of both queues very accurately in the mean and standard deviation. Communication link $N_2 \rightarrow N_3$ is faster than combined service rate of links $N_0 \rightarrow N_2$ and $N_1 \rightarrow N_2$



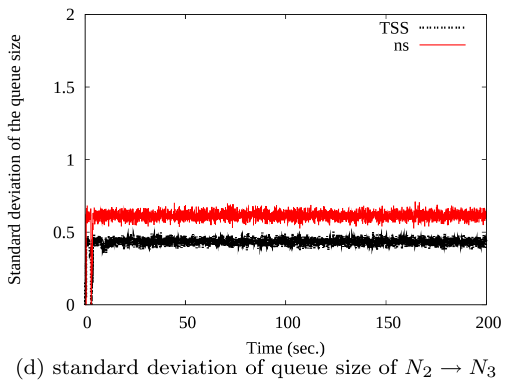
(a) mean queue size of $N_0 \rightarrow N_2$



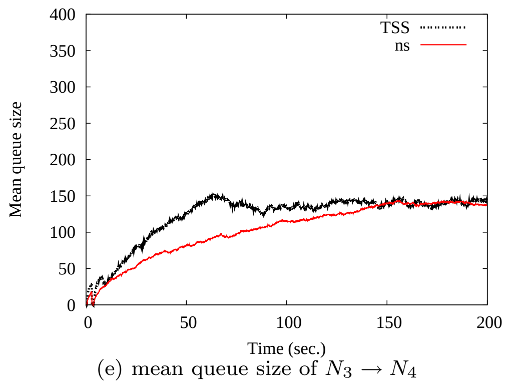
(b) standard deviation of queue size of $N_0 \rightarrow N_2$



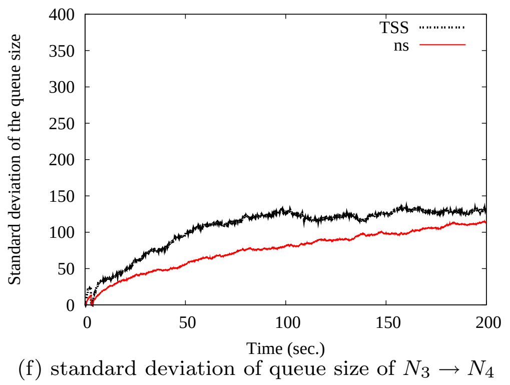
(c) mean queue size of $N_2 \rightarrow N_3$



(d) standard deviation of queue size of $N_2 \rightarrow N_3$

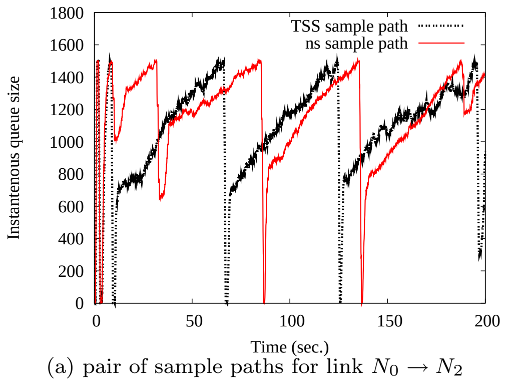


(e) mean queue size of $N_3 \rightarrow N_4$

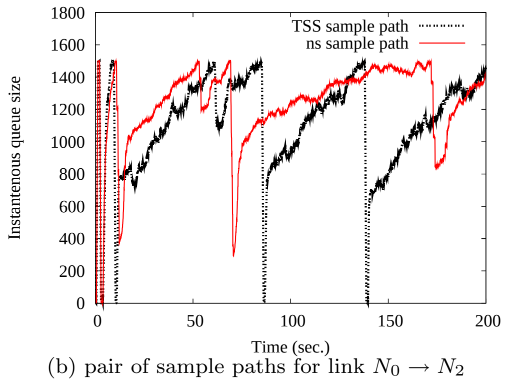


(f) standard deviation of queue size of $N_3 \rightarrow N_4$

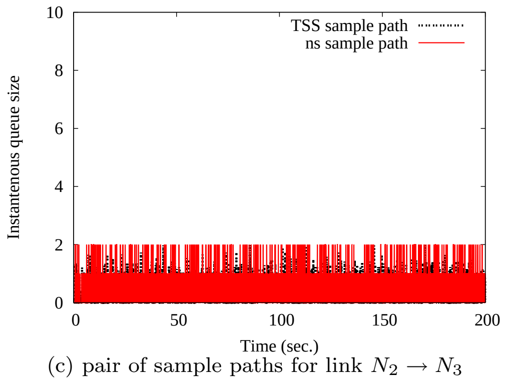
Figure 6.12: Comparison of simulation results with TSS for Scenario 1 for the six-node network in Figure 6.11.



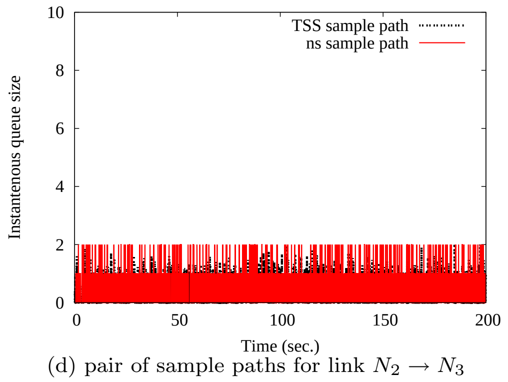
(a) pair of sample paths for link $N_0 \rightarrow N_2$



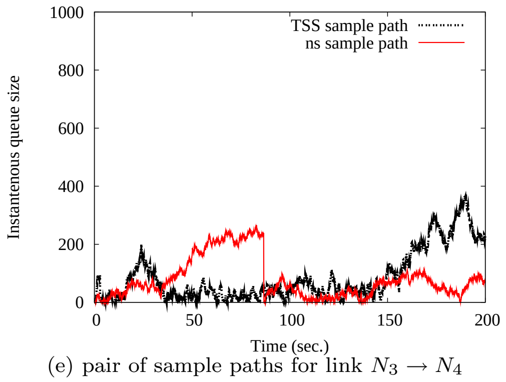
(b) pair of sample paths for link $N_0 \rightarrow N_2$



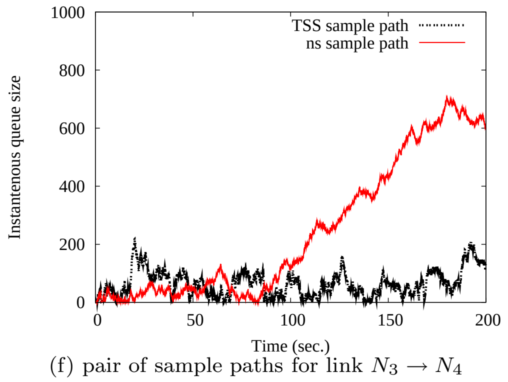
(c) pair of sample paths for link $N_2 \rightarrow N_3$



(d) pair of sample paths for link $N_2 \rightarrow N_3$



(e) pair of sample paths for link $N_3 \rightarrow N_4$



(f) pair of sample paths for link $N_3 \rightarrow N_4$

Figure 6.13: Comparison of sample paths generated by packet-level simulation with results obtained using TSS in Scenario 1 for the six-node network in Figure 6.11.

thus the queue on this link is very small.

Even in this case TSS approximation is very close to the packet-level simulation (Figures 6.12c and 6.12d). Figure 6.13 shows pairs of sample paths as generated by packet-level simulation and TSS. Especially interesting are the sample paths of link $N_3 \rightarrow N_4$ presented on Figures 6.13e and 6.13f. Behavior like this, when a queue has utilization very close to 1 and the sample paths diversity is caused by randomness of service and arrival processes, can not be captured by methods such as fluid approximation. Furthermore, capturing it is important because the queuing delay on such link is considerable and affects the behavior of its TCP flows.

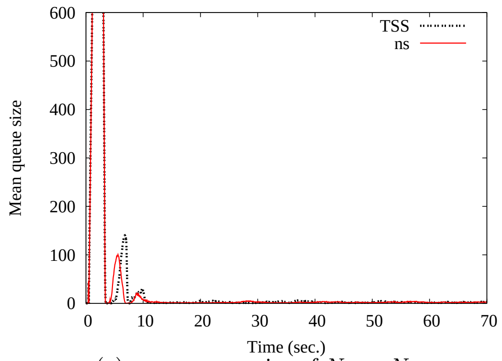
Scenario 2

This scenario presents TSS accuracy in case of a single bottleneck link. Service rates and coefficients of variation for all links are the same as in Scenario 1 except that link $N_2 \rightarrow N_3$ has service rate of 3000 packets/sec which is less than the maximum aggregate departure from links $N_0 \rightarrow N_2$ and $N_1 \rightarrow N_2$. Thus link $N_2 \rightarrow N_3$ is the bottleneck. Traffic is generated by 20 TCP flows, 10 from nodes N_0 to N_4 and 10 from nodes N_1 to N_5. Buffering capacity of each link is 1500 packets.

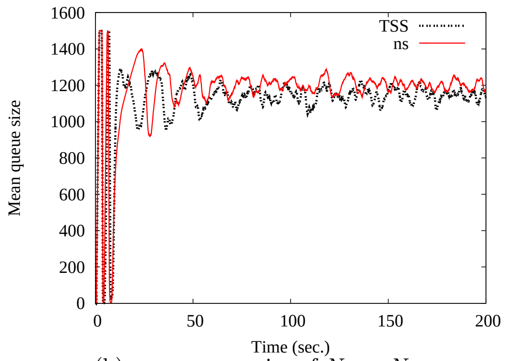
Figure 6.14 presents the time evolution of mean queue size (a and b) and its standard deviation (c and d) for links $N_0 \rightarrow N_2$ and $N_2 \rightarrow N_3$, respectively. TSS approximation is very accurate for both the mean and the standard deviation. Figure 6.15 presents pairs of samples paths for links $N_0 \rightarrow N_2$ and $N_2 \rightarrow N_3$ as computed by packet-level simulator and TSS.

6.4 Small network

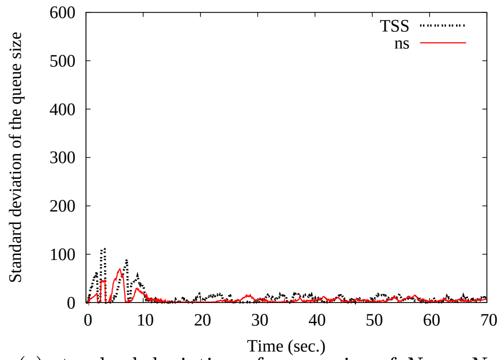
Next example configuration consists of 14 nodes and 13 communication links. The central link ($N_{12} \rightarrow N_{13}$) has service rate of 2000 packets/sec and all other



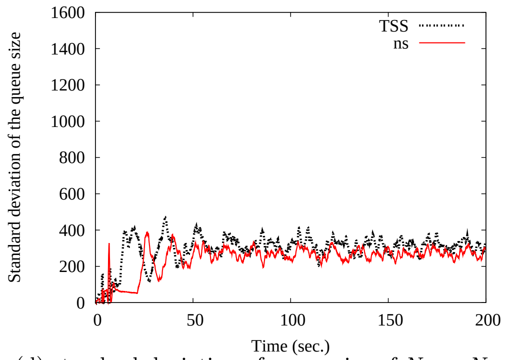
(a) mean queue size of $N_0 \longrightarrow N_2$



(b) mean queue size of $N_2 \longrightarrow N_3$

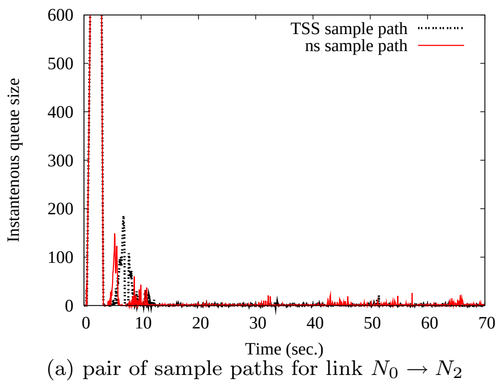


(c) standard deviation of queue size of $N_0 \longrightarrow N_2$

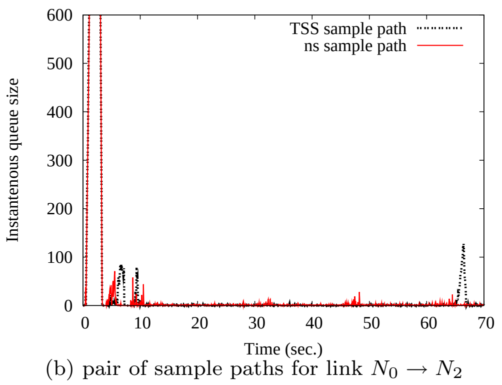


(d) standard deviation of queue size of $N_2 \longrightarrow N_3$

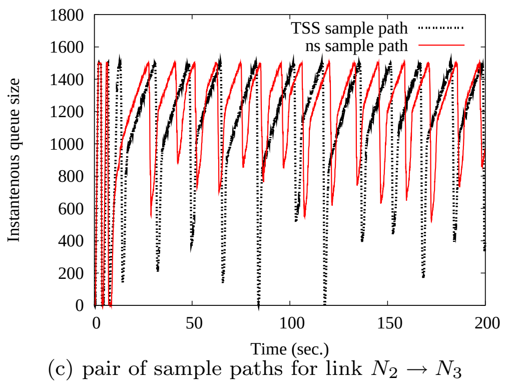
Figure 6.14: Comparison of simulation results with TSS for Scenario 2 for the six-node network in Figure 6.11.



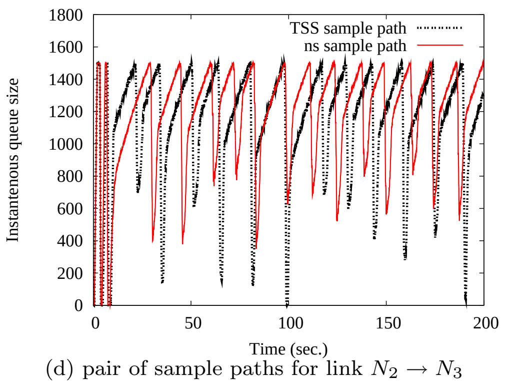
(a) pair of sample paths for link $N_0 \to N_2$



(b) pair of sample paths for link $N_0 \to N_2$



(c) pair of sample paths for link $N_2 \to N_3$



(d) pair of sample paths for link $N_2 \to N_3$

Figure 6.15: Comparison of sample paths generated by packet-level simulation with results obtained using TSS in Scenario 2 for the six-node network in Figure 6.11.

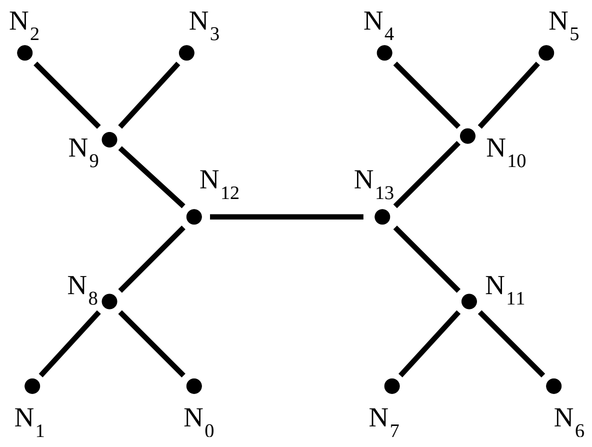
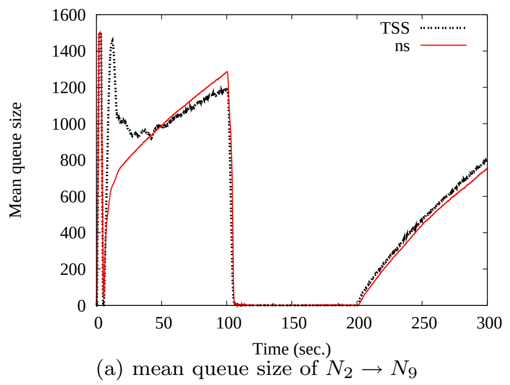


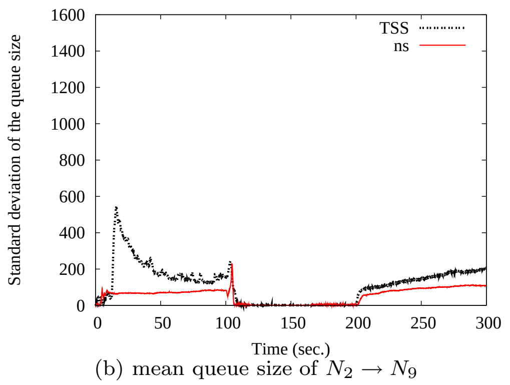
Figure 6.16: Small network topology.

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 till 300 second. 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 second till 200 second.

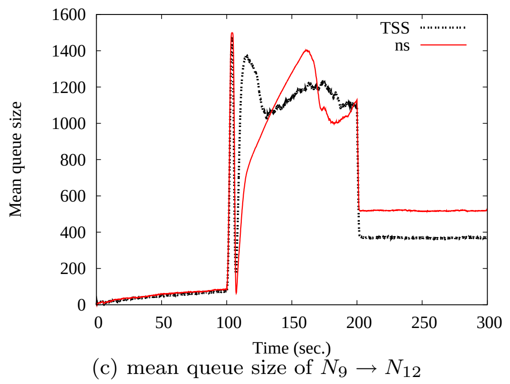
Figures 6.17 and 6.18 present comparison of mean queue size and its deviation as computed using packet-level simulation and TSS. Figures 6.17a and 6.17b present mean and standard deviation for link $N_2 \rightarrow N_9$. Figures 6.17c and 6.17d present mean and standard deviation for link $N_9 \rightarrow N_{12}$. Figures 6.17e and 6.17f present mean and standard deviation for link $N_{12} \rightarrow N_{13}$. Figures 6.18a and 6.18c present mean and standard deviation for link $N_0 \rightarrow N_8$. Figures 6.18b and 6.18d present mean and standard deviation for link $N_8 \rightarrow N_{12}$. TSS correctly predicts periods of large queuing for all links of interest. It also approximates variability of the queue size very well. Finally Figures 6.19a and 6.19b present



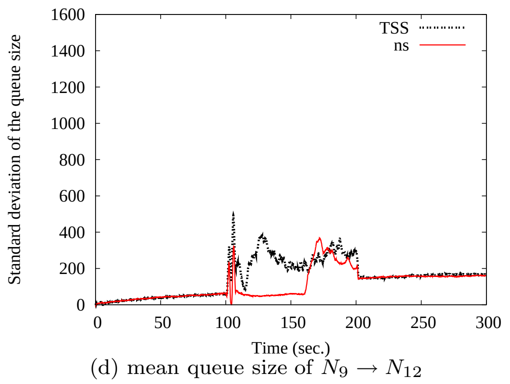
(a) mean queue size of $N_2 \longrightarrow N_9$



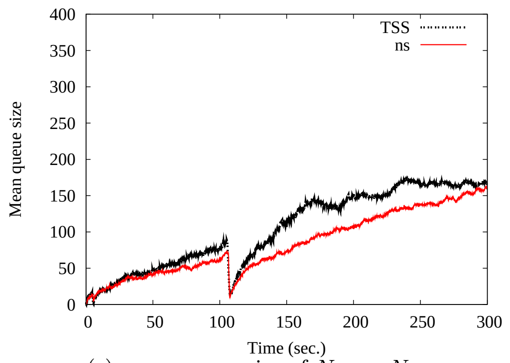
(b) mean queue size of $N_2 \longrightarrow N_9$



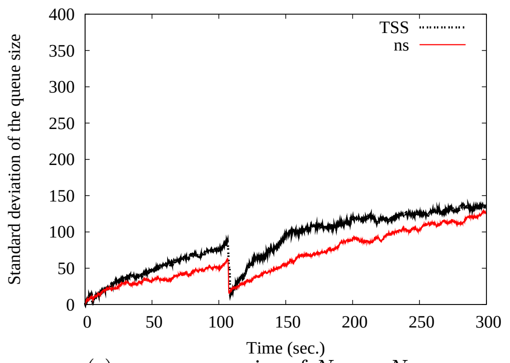
(c) mean queue size of $N_9 \longrightarrow N_{12}$



(d) mean queue size of $N_9 \longrightarrow N_{12}$

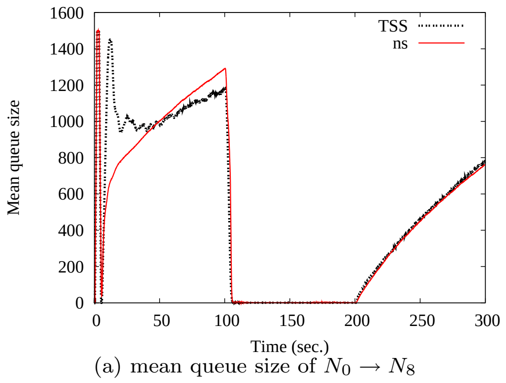


(e) mean queue size of $N_{12} \longrightarrow N_{13}$

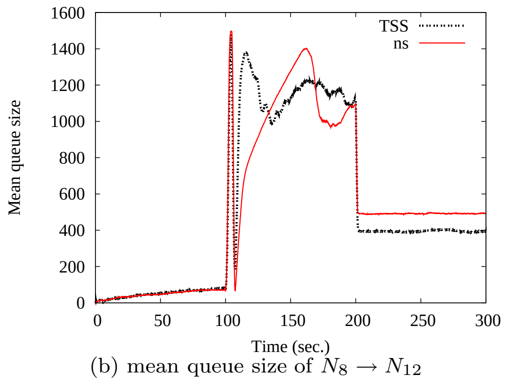


(e) mean queue size of $N_{12} \longrightarrow N_{13}$

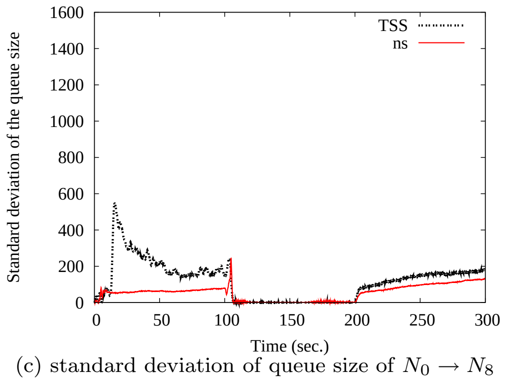
Figure 6.17: Comparison of mean queue size and its standard deviation for links $N_2 \longrightarrow N_9$, $N_9 \longrightarrow N_{12}$, and $N_{12} \longrightarrow N_{13}$ for network in Figure 6.16.



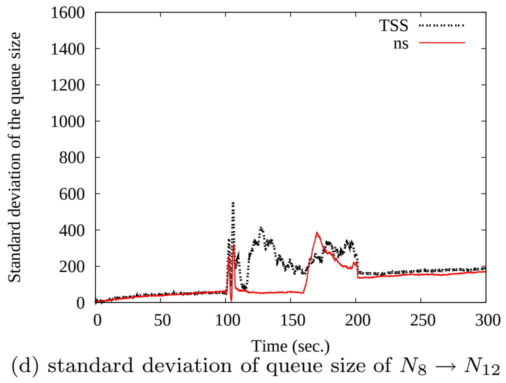
(a) mean queue size of $N_0 \rightarrow N_8$



(b) mean queue size of $N_8 \rightarrow N_{12}$



(c) standard deviation of queue size of $N_0 \rightarrow N_8$



(d) standard deviation of queue size of $N_8 \rightarrow N_{12}$

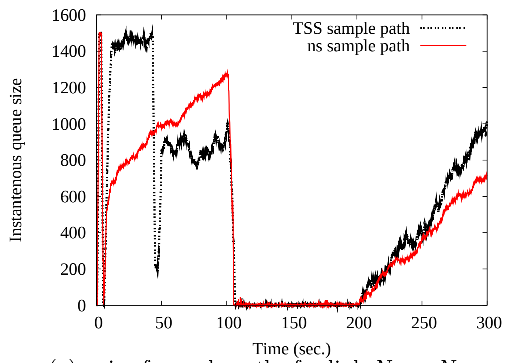
Figure 6.18: Comparison of mean queue size and its standard deviation for links $N_0 \rightarrow N_8$ and $N_8 \rightarrow N_{12}$ for network in Figure 6.16.

two pairs of sample paths for link $N_0 \rightarrow N_8$. Similarly, figures 6.19c, 6.19d, and figures 6.19e, 6.19f for links $N_8 \rightarrow N_{12}$ and $N_{11} \rightarrow N_7$., respectively.

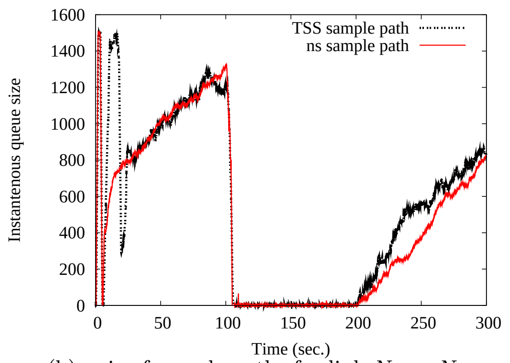
6.5 Large network

This section examines the quality of TSS approximation for a 35-node network with 35 communication links and 180 TCP flows shown in Figure 6.20. All links have service rate of 5000 packets/sec, which corresponds to (assuming 1KB packets) approximately 40 Mbps. Buffer capacities of communication links are 1000 packets for backbone ring (i.e., links connecting ring of nodes N_0, N_1, N_2, N_3, N_4, N_5, N_6), and 5000 packets for the remaining links. 10 bulk TCP flows are continuously active for each of the following source-destinations: $N_{14} \rightarrow N_{17}$, $N_{15} \rightarrow N_{18}$, $N_{16} \rightarrow N_{19}$, $N_{14} \rightarrow N_{23}$, $N_{15} \rightarrow N_{24}$, $N_{16} \rightarrow N_{25}$, $N_{20} \rightarrow N_{23}$, $N_{21} \rightarrow N_{24}$, $N_{22} \rightarrow N_{25}$, $N_{20} \rightarrow N_{26}$, $N_{21} \rightarrow N_{27}$, $N_{22} \rightarrow N_{28}$, $N_{29} \rightarrow N_{32}$, $N_{30} \rightarrow N_{33}$, $N_{31} \rightarrow N_{34}$, $N_{29} \rightarrow N_{17}$, $N_{30} \rightarrow N_{18}$, $N_{31} \rightarrow N_{19}$.

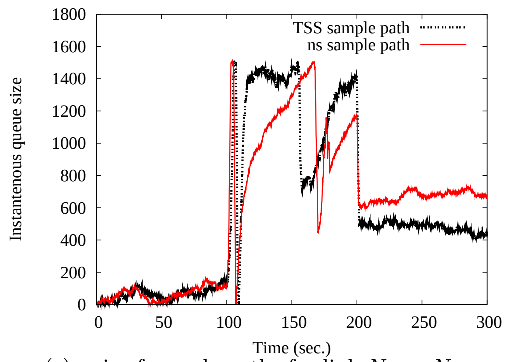
Because this configuration has multiple interacting TCP flows on a non-trivial topology, it gives raise to various behaviors and is a good configuration for testing the quality of TSS approximation. Figure 6.21 shows time evolution of metrics of interest for link $N_0 \rightarrow N_1$. Figures 6.21a and 6.21b present time evolution of mean queue size and its standard deviation, respectively. TSS results match nearly exactly those of packet-level simulation. Figures 6.21c and 6.21d show two pairs of sample paths for this link as generated by TSS and the simulator. Finally, figures 6.21e and 6.21f are magnifications of figures 6.21c and 6.21d, respectively. Corresponding results (without sample paths magnification) for links $N_{12} \rightarrow N_5$, $N_2 \rightarrow N_3$ and, $N_7 \rightarrow N_0$ are presented in Figures 6.22, 6.23, and 6.24, respectively.



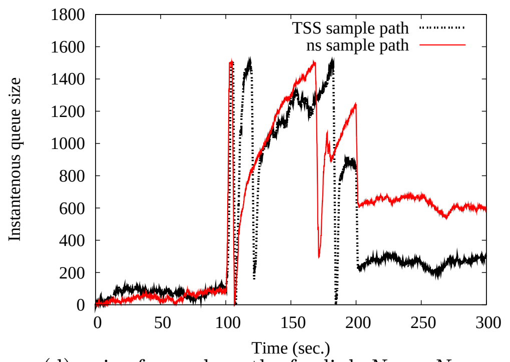
(a) pair of sample paths for link $N_0 \rightarrow N_8$



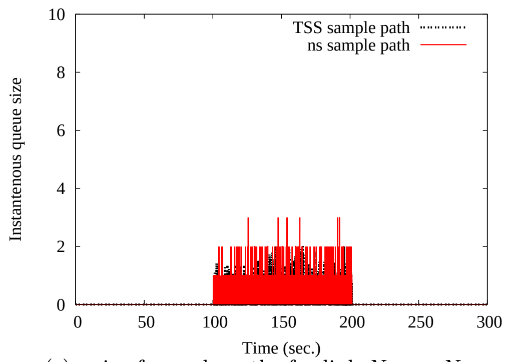
(b) pair of sample paths for link $N_0 \rightarrow N_8$



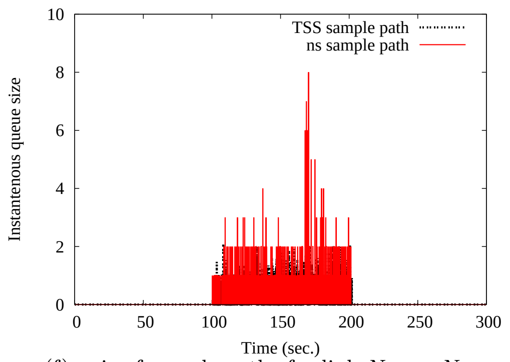
(c) pair of sample paths for link $N_8 \rightarrow N_{12}$



(d) pair of sample paths for link $N_8 \rightarrow N_{12}$



(e) pair of sample paths for link $N_{11} \rightarrow N_7$



(f) pair of sample paths for link $N_{11} \rightarrow N_7$

Figure 6.19: Pairs of queue size sample paths for links $N_0 \rightarrow N_8$, $N_8 \rightarrow N_{12}$, and $N_{11} \rightarrow N_7$ for network configuration in Figure 6.16.

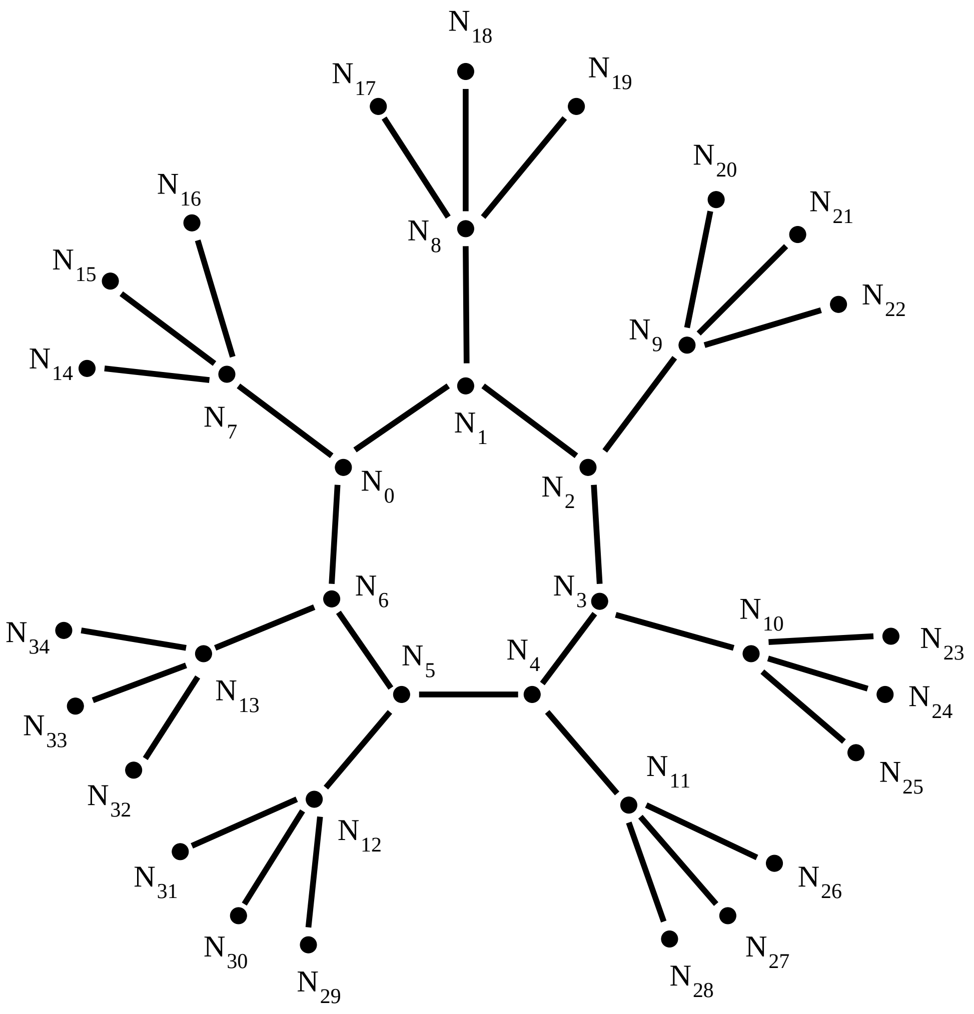
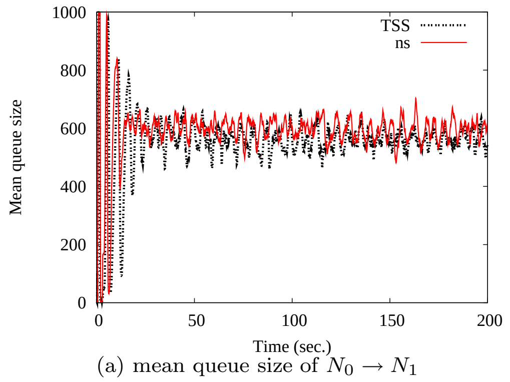
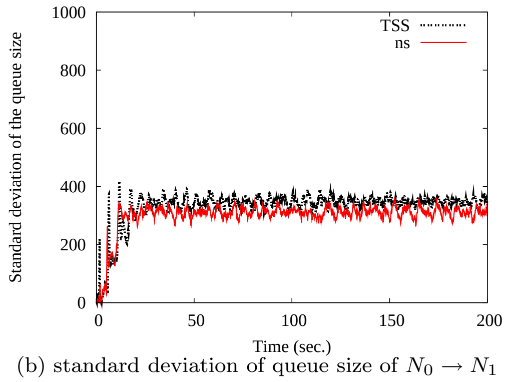


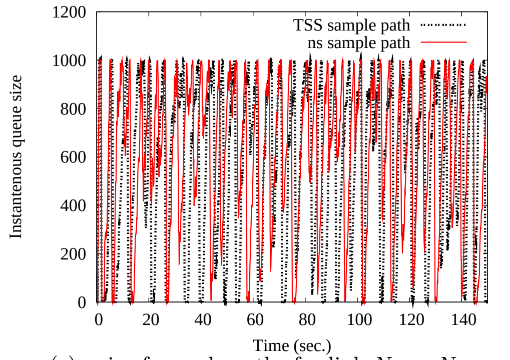
Figure 6.20: Large network topology: 35-nodes, 35 communication links, and 180 TCP flows.



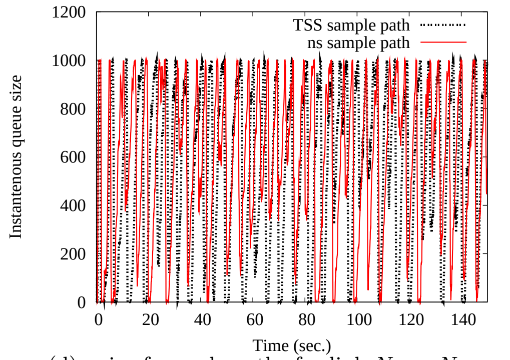
(a) mean queue size of $N_0 \rightarrow N_1$



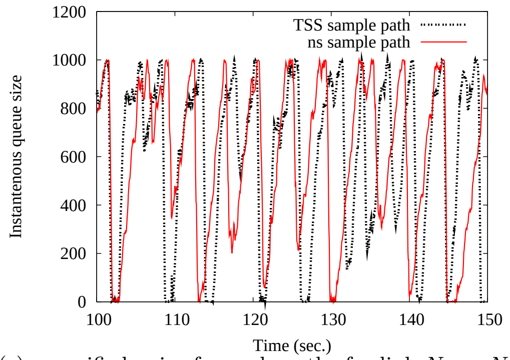
(b) standard deviation of queue size of $N_0 \rightarrow N_1$



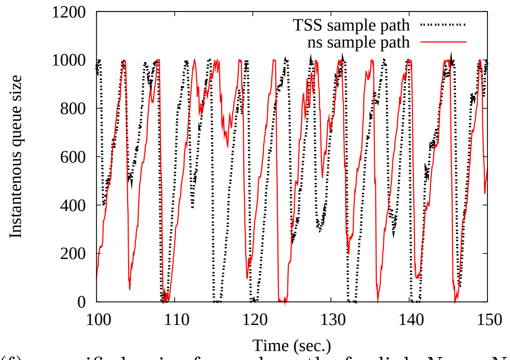
(c) pair of sample paths for link $N_0 \rightarrow N_1$



(d) pair of sample paths for link $N_0 \rightarrow N_1$

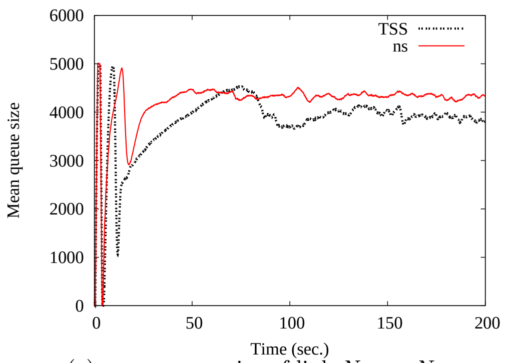


(e) magnified pair of sample paths for link $N_0 \rightarrow N_1$

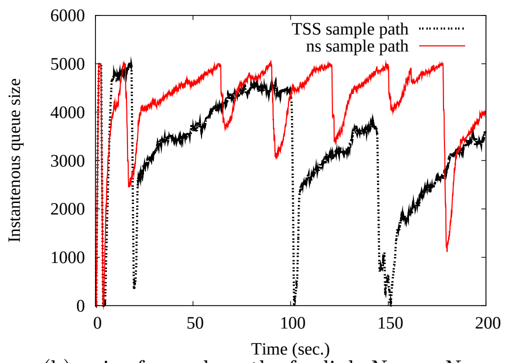


(f) magnified pair of sample paths for link $N_0 \rightarrow N_1$

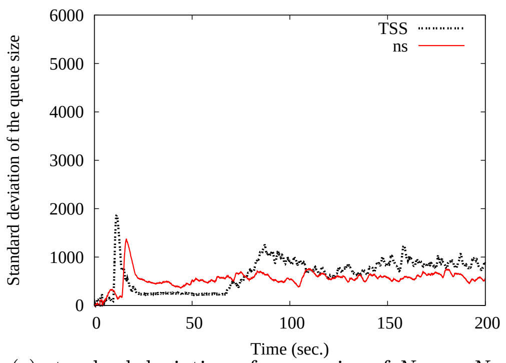
Figure 6.21: Comparison of mean queue size, its standard deviation, and sample paths as computed by packet-level simulator and TSS for link $N_0 \rightarrow N_1$ from network in Figure 6.20.



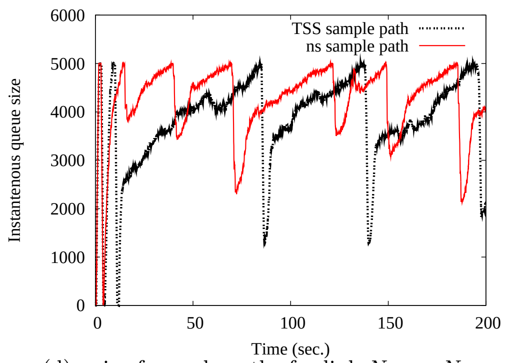
(a) mean queue size of link $N_{12} \rightarrow N_5$



(b) pair of sample paths for link $N_{12} \rightarrow N_5$

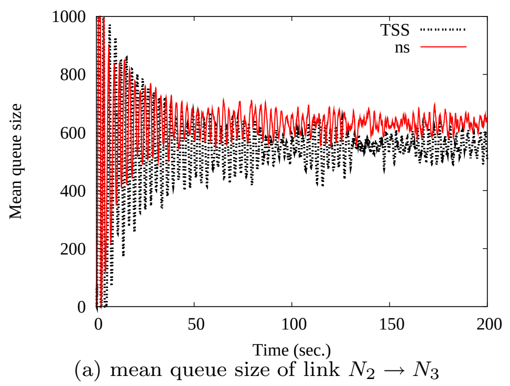


(c) standard deviation of queue size of $N_{12} \rightarrow N_5$

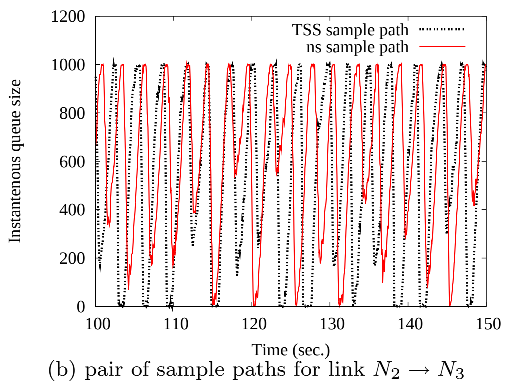


(d) pair of sample paths for link $N_{12} \rightarrow N_5$

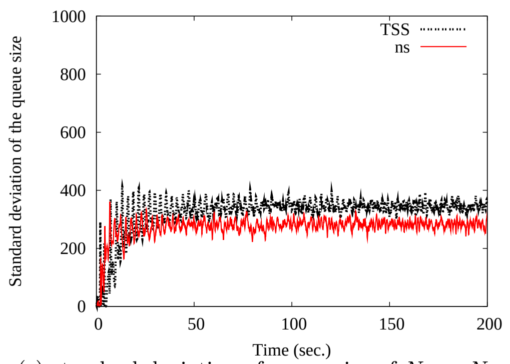
Figure 6.22: Comparison of mean queue size, its standard deviation, and sample paths as computed by packet-level simulator and TSS for link $N_{12} \rightarrow N_5$ for network in Figure 6.20.



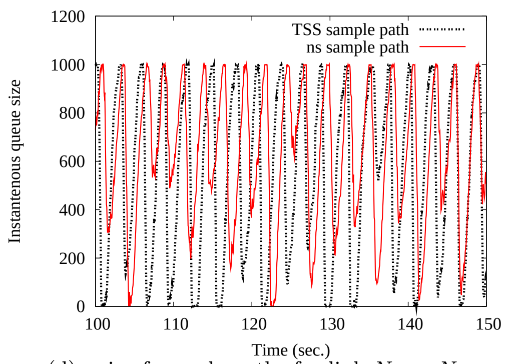
(a) mean queue size of link $N_2 \rightarrow N_3$



(b) pair of sample paths for link $N_2 \rightarrow N_3$

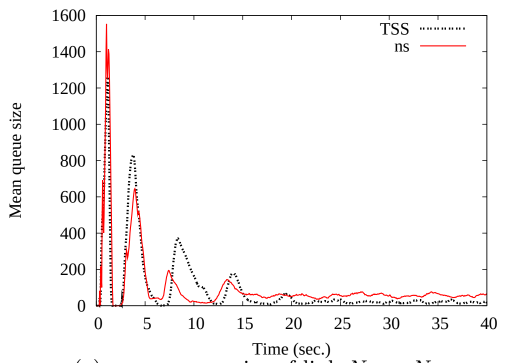


(c) standard deviation of queue size of $N_2 \rightarrow N_3$

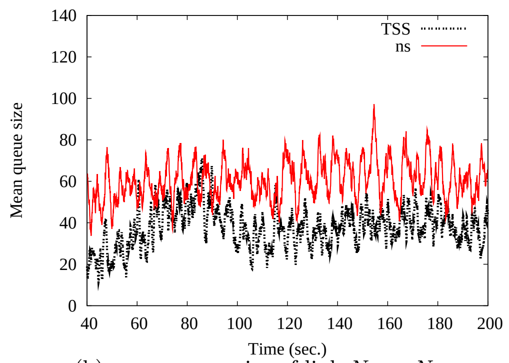


(d) pair of sample paths for link $N_2 \rightarrow N_3$

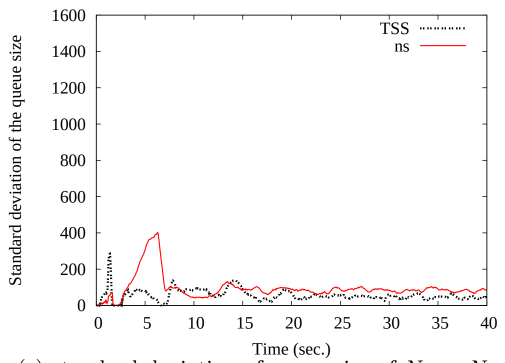
Figure 6.23: Comparison of mean queue size, its standard deviation, and sample paths as computed by packet-level simulator and TSS for link $N_2 \rightarrow N_3$ for network in Figure 6.20.



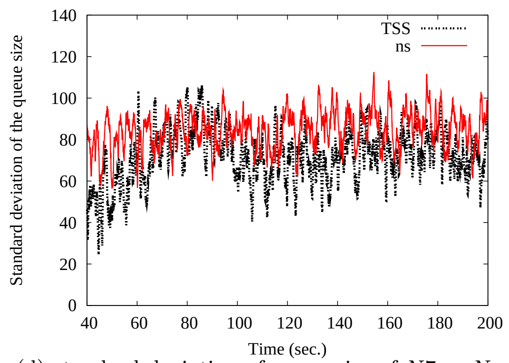
(a) mean queue size of link $N_7 \rightarrow N_0$



(b) mean queue size of link $N_7 \rightarrow N_0$



(c) standard deviation of queue size of $N_7 \rightarrow N_0$



(d) standard deviation of queue size of $N7 \rightarrow N_0$

Figure 6.24: Comparison of mean queue size and its standard deviation as computed by packet-level simulator and TSS for link $N_7 \longrightarrow N_0$ (with magnifications) for network in Figure 6.20.

Chapter 7

Conclusions and Future Directions

Performance evaluation of computer networks is an important and challenging task. It is needed not only to design new protocols and test their efficiency, but also to manage and optimize existing systems. Yet existing evaluation techniques are not applicable to modern computer networks. The simplifying assumptions required for analytical models make them applicable only to the simplest networks, such as networks where all arrival and service processes are Markovian. Packet-level simulation, probably the most popular evaluation method, can be very accurate but it has very high computational cost, and hence applicable only to small networks and low-speed communication links. Other methods, for instance fluid approximation, are not applicable to networks with general state-dependent control, such as TCP's congestion control.

This Dissertation presented Timestep Stochastic Simulation for fast generation of sample paths of computer networks. We first introduced the method for a case of single communication link. The time evolution of the queue size is divided into steps of size δ chosen to be smaller than the reaction time of the congestion control protocol (e.g., TCP round-trip time). 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. Both the arrival and service processes are represented using first two moments of approximating renewal processes, assumed to be stationary within each timestep.

The method is much faster than packet-level simulation and has almost the same accuracy. Because state-dependent control feedback is based on sample path metrics, our method is more suitable for modeling state-dependent control schemes (such as TCP's) than fluid approximations. Since the transmission and reception events are replaced with timesteps, TSS is not affected by the increase of link bandwidth, as is the case with packet-level simulation. TSS also handles well communication links with utilization close to 1 for large periods of time. This is an extremely difficult case that other approaches do not handle. Queues with this property (which is common in networks of links with equal transmission rates) are ignored by fluid approximation, while in reality they contribute to queuing delays and significantly affect the behavior of the network.

We then extended the method to handle networks of queues. The issue here is to handle the internal flows, i.e., the departures processes of queues, and the processes resulting from splitting and merging flows. These internal flows are in general not renewal processes, and thus exact analytical formulas for moments of internal flows are not available. We represent them using first two moments of a suitable approximating renewal process. Again, we assume that the first two moments are constant throughout each timestep δ.

TSS is a modeling framework in which general state-dependent congestion control mechanisms may be represented. The flow model in TSS is time-stepped, i.e., the state of the traffic source can be adjusted only at timesteps every δ. The adjustment may be based on the history of the sample state of the network, such as round-trip time or loss rate averaged over last second, which makes our method suitable for general state-dependent control. We illustrated applicability of the TSS by presenting traffic models for time-dependent UDP flows, state-dependent UDP flows, and TCP flows.

We validated the quality of approximation by comparing results obtained by TSS against the corresponding results obtained using packet-level simulation. TSS generates diverse sample paths that are very close (in the sense of their distributions) to the ones generated by packet-level simulator for a fraction of the computational cost. Averaging over multiple repetitions of both the TSS and the packet-level simulation shows excellent agreement of the time evolution of the first and second moments of queue size. It is also possible to obtain higher order moments and other metrics of interest, such as probability of loss.

Possible future research involves applications of the method to various networking problems. One interesting problem is optimal placement of caches in a network. TSS can be used to quickly test proposed configuration, thus enabling automation of the placement process. Another interesting problem is dynamic routing and load balancing. In this case TSS may be used by the research community to evaluate the efficiency of proposed routing mechanisms or balancing schemes.

BIBLIOGRAPHY

[1] M. Ammar et. al, "Simulation of Large-Scale Communication Networks How Large? How Fast?," in *MASCOTS*, 2003.

[2] "Cisco 1200 series router," http://www.cisco.com/.

[3] "Qualnet," http://www.scalable-networks.com.

[4] L. Kleinrock, *Queueing systems volume I*, John Wiley and Sons, 1976.

[5] L. Kleinrock, *Queueing systems volume II*, John Wiley and Sons, 1976.

[6] 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, pp. 151–160.

[7] Y. Liu, F. Lo Presti, V. Misra, D. Towsley, and Y. Gu, "Fluid Models and Solutions for Large-Scale IP Networks," in *ACM SIGMETRICS*, June 2003, pp. 91–101.

[8] 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 *ACM SIGMETRICS*, June 2003, pp. 58–69.

[9] A. Kolmogorov, "Ueber die analytischen Methoden in der Wahrscheinlichkeitsrechnung," in *Mathematical Annals*, 1931, vol. 104, pp. 415–458.

[10] W. Feller, "Diffusion Processes in One Dimension," in *Transactions of the American Mathematical Society*, July 1954, vol. 77, pp. 1–31.

[11] "The Network Simulator - ns-2," http://www.isi.edu/nsnam/ns.

[12] M. Neuts, *Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach*, The Johns Hopkins University Press, 1981.

[13] L. Lipsky, *Queueing theory : a linear algebraic approach*, Macmillan Publishing Company, 1992.

[14] G. Newell, *Applications of queueing theory*, Chapman and Hall, 1982.

[15] Ch. Knessl and Y. P. Yang, "An Exact Solution for an $M(t)/M(t)/1$ Queue with Time-Dependent Arrivals and Service," in *Queueing Systems*, April 2002, vol. 40, pp. 233–245.

[16] W. Whitt, "The Queueing Network Analyzer," in *Bell System Technical Journal*, November 1983, vol. 62, pp. 2779–2815.

[17] W. Whitt, "Performance of the Queueing Network Analyzer," in *Bell System Technical Journal*, November 1983, vol. 62, pp. 2817–2843.

[18] W. Kraemer and M. Langenbach-Belz, "Approximate Formulae for the Delay in the Queueing System GI/G/1," in *Congressbook of Eighth International Teletrafic Congress*, 1976.

[19] S. Albin, "Approximating a Point Process by a Renewal Process, II: Superposition Arrival Processes to Queues," in *Operations Research*, 1984, pp. 1133–1162.

[20] K. Sunkyo, "The heavy-traffic bottleneck phenomenon under splitting and superposition," in *European Journal of Operational Research*, 2004, pp. 736–745.

[21] W. Whitt, "Towards Better Multi-Class Parametric-Decomposition Approximations for Open Queueing Networks," in *Annals of Operations Research*, 1994, vol. 48, pp. 221–248.

[22] H. Kobayashi, "Application of the Diffusion Approximation to Queueing Networks I: Equilibrium Queue Distributions," in *ACM (JACM)*, April 1974, vol. 21, pp. 316–328.

[23] H. Kobayashi, "Application of the Diffusion Approximation to Queueing Networks II: Nonequilibrium Distributions and Applications to Computer Modeling," in *ACM (JACM)*, July 1974, vol. 21, pp. 459–469.

[24] E. Gelenbe, "On Approximate Computer System Models," in *ACM (JACM)*, April 1975, vol. 22, pp. 261–269.

[25] E. Gelenbe, "Probabilistic Models of Computer Systems, Part II: Diffusion Approximation, Waiting Times and Batch Arrivals," 1979.

[26] P.J. Kuehn, "Approximate Analysis of a General Queuing Networks by Decomposition," in *IEEE Transactions on Communications*, 1979, vol. 27, pp. 113–126.

[27] A. Duda, "Diffusion approximation for time-dependent queueing systems," in *IEEE JSAC*, 1986, vol. SAC-4, pp. 905–918.

[28] A. Duda, "Transient diffusion approximation for some queueing systems," in *ACM SIGMETRICS*, 1983, pp. 118–128.

[29] Ibrahim Matta and A. Udaya Shankar, "Z-Iteration: A Simple Method for Throughput Estimation in Time-Dependent Multi-Class Systems," in *Measurement and Modeling of Computer Systems*, 1995, pp. 126–135.

[30] C. T. Popescu and A. U. Shankar, "Fast Evaluation of Ensemble Transients of $M(t)/M(t)/.$ Networks," December 1999, http://www.cs.umd.edu/ shankar/Z-iteration/.

[31] V. Misra, W. Gong, and D. Towsley, "Stochastic Differential Equation Modeling and Analysis of TCP-Windowsize Behavior," in *Proceedings of Sigmetrics Performance*, 1999.

[32] Y. Guo, W. Gong, and D. Towsley, "Time-stepped Hybrid Simulation (TSHS)For Large Scale Networks," in *INFOCOM*, 2000, pp. 441–450.

[33] G. F. Newell, "Queues with Time-Dependent Arrival Rates: I - The Transition Through Saturation," in *Journal of Applied Probability*, 1968, vol. 5, pp. 436–451.

[34] G.F. Newell, "Queues with time-dependent arrival rates II: The Maximum Queue and the Return to Equilibrium," in *Journal of Applied Probability*, 1968, vol. 22, pp. 579–590.

[35] G. F. Newell, "Queues with time-dependent arrival rates III: The mild rush hour," in *Journal of Applied Probability*, 1968, vol. 5, pp. 591–606.

[36] D. P. Gaver, "Diffusion Approximations and Models for Certain Congestion Problems," in *Journal of Applied Probability*, 1968, vol. 5, pp. 607–623.

[37] J. F. C. Kingman, "The heavy traffic approximation in the theory of queues," in *Proceedings of the Symposium on Congestion Theory (W. L. Smith et al., eds.) 137159. Univ. North Carolina Press, Chapel Hill.*, 1965, pp. 137–169.

[38] K. T. Marshall, "Some Relationships Between the Distributions of Waiting Time, Idle Time and Interoutput Time in the $GI/G/1$ Queue," *SIAM Journal of Applied Mathematics*, vol. 16, no. 2, pp. 324–327, March 1968.

[39] W. Whitt, "Approximations for Departure Processes and Queues in Series," in *Naval Research Logistics Quarterly*, December 1984, vol. 31, pp. 499–521.

[40] J. Fraker, "Approximate Techniques for the Analysis of Tandem Queueing Systems," in *Ph.D. dissertation, Department of Industrial Engineering, Clemson University*, 1971.

[41] H. Stehfest, "Algorithm 368. Numerical inversion of Laplace transforms," in *Communications of the ACM*, 1970, vol. 13, pp. 47–49.

[42] "RFC 768 - User Datagram Protocol," 1980.

[43] "RFC 793 - Transmission Control Protocol," 1981.

[44] "RFC 2001 - TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," 1997.

[45] "RFC 2582 - The NewReno Modification to TCP's Fast Recovery Algorithm," 1999.

[46] Lawrence S. Brakmo, Sean W. O'Malley, and Larry L. Peterson, "TCP Vegas: New Techniques for Congestion Detection and Avoidance," in *SIGCOMM*, 1994, pp. 24–35.

[47] D. R. Cox, "A use of complex probabilities in the theory of stochastic processes," in *Proceedings of the Cambridge Philosophical Society*, 1955, vol. 51, pp. 313–319.

[48] D. R. Cox and H. D. Miller, *The Theory of Stochastic Processes*, Methuen, 1965.

[49] W. Feller, "The parabolic differential equations and the associated semi-groups of transformations," in *Annals of Mathematics*, 1952, vol. 55, pp. 468–519.

[50] W. Whitt, "On Approximations for Queues, I: Extremal Distributions," in *AT&T Bell Laboratories Technical Journal*, 1984, pp. 115–138.

[51] J. Klincewicz and W. Whitt, "On Approximations for Queues, II: Shape Constraints," in *AT&T Bell Laboratories Technical Journal*, 1984, pp. 139–161.

[52] W. Whitt, "On Approximations for Queues, III: Mixtures of Exponential Distributions," in *AT&T Bell Laboratories Technical Journal*, 1984, pp. 163–175.

[53] W. Whitt, "Approximating a Point Process by a Renewal Process: Two Basic Methods," in *Operations Research*, 1982, vol. 30, pp. 125–147.