Network Invariant-Based Fast Simulation for Large Scale TCP/IP Networks

Hwangnam Kim, Hyuk Lim, and Jennifer C. Hou
Department of Computer Science
University of Illinois at Urbana-Champaign
201 N. Goodwin, Urbana, IL 61801
Email: {hkim27, hyuklim, jhou}@cs.uiuc.edu

ABSTRACT

In this paper, we present a rescaling simulation methodology (RSM) to expedite simulation in large-scale TCP/IP networks without loss of fidelity of simulation results. Conceptually, we scale down the network to be simulated to reduce the number of events, simulate the downscaled network for a short period of time, and then extrapolate the corresponding results for the original network by scaling up the simulation results obtained from the downscaled network. Both the operations of scaling down and rescaling up the network are conducted in such a manner that the network invariant, called the *bandwidth-delay* product, is preserved. In particular, since the dynamics of queues, such as the queue size, the dropping probability, and other parameters at every link are the same in both the original and downscaled network, RSM can accurately infer the network dynamics behavior of the original network (equipped with various Active Queue Management (AQM) strategies). In contrast to SHRiNK [17], RSM does not make any assumption on the input traffic and can work with any AQM strategy. It also preserves the queue dynamics and the network capacity as perceived by TCP connections.

To validate the proposed methodology, we have implemented RSM based simulation in ns-2, and conducted a simulation study comparing RSM based simulation against packet level simulation, with respect to the capability of capturing transient, packet-level network dynamics, the execution time and the discrepancy in simulation results. The simulation results indicate an order of magnitude or more improvement (maximally 50 times) in execution time and the performance improvement becomes more prominent as the network size increases (in terms of number of nodes and network capacity) or as the scaling parameter decreases. The error discrepancy between RSM based simulation and packet level simulation, on the contrary, is minimally 1-2 % and maximally 10 % in a wide variety of network topologies (with various AQM strategies) and traffic loads. The encouraging simulation results, coupled with the fact that implementation of RSM is simple and straightforward, suggest that RSM can be used to simulate, and accurately infer network dynamics of, large-scale TCP/IP networks.

Keywords

performance modeling and simulation, rescaling simulation methodology, large scale networks, AQM

1. INTRODUCTION

Modern data communication networks are extremely complex and do not lend well to theoretical analysis. With computer/network entities and techniques interacting and interfering with one another, optimization problems do not have a simple and regular structure that allows us to neatly fit it into the framework of established optimization theories. As a result, it may be more feasible to carry out simulation to study and evaluate the performance of network entities and protocols, and interaction among them. The major obstacle in packet-level network simulation is, however, the vast number of packets that have to be simulated in order to produce accurate results, especially in large-scale networks. Each packet will generate a number of events (e.g., arrival of a packet at the router, its departure, and its queuing, just to name a few) on the path from the source to the destination and each event has to be executed at some specified time point. As the CPU time required is roughly proportional to the number of events that have to be executed, packet-level simulation easily becomes computationally expensive, if not infeasible, when the network size and/or the traffic amount is extremely large. What seems to be a reasonable solution is really to combine theoretical modeling with packet-level simulation [14, 16, 19].

The notion of *fluid model based simulation* is proposed to reduce the computational load in packet level simulation [8, 11, 13, 19, 21]. Conceptually a fluid model — a set of differential equations that characterize the network dynamics — is adopted and incorporated into the simulation engine. In the course of simulation, a sequence of closely-spaced packets is abstracted into a fluid chunk (usually characterized by the fluid rate) and the fluid model is used to obtain the parameters of interest (e.g., the system throughput). In spite of its effectiveness in terms of reducing the execution time, fluid model based simulation is not well-suited for studying the network behavior under light and/or sporadic traffic, as it is built upon the assumption of existence of a large number of active flows in the network (so that the composite traffic can be modeled as a flow). To deal with this issue, *network calculus*-based simulation was proposed [9]. Kim and Hou [9] characterize how TCP congestion control — additive increase and multiplicative decrease (AIMD) — interacts with AQM strategies with network calculus theory

[1, 3, 5], derive upper and lower bounds on the attainable TCP throughput, incorporate the models in *ns-2*, and then instrument the simulator to regulate TCP flows in compliance with the model. Although network calculus-based simulation indeed gives encouraging results, it cannot provide the packet level dynamics, such as the instantaneous queue length and packet dropping probability, due to the use of network calculus. (Note that fluid model based simulation also suffers from this shortcoming.)

In this paper, we take a dramatic departure from the above approaches and propose a new rescaling simulation methodology (RSM) for simulating large-scale TCP/IP networks. As mentioned above, as the number of events increases with the network size and the amount of traffic, the computational cost increases accordingly. If we can scale down the network to one that can be simulated at the packet level in a short time interval to produce sufficient results and extrapolate, without loss of accuracy, results for the original network, we can significantly reduce the computational cost and yet preserve the network dynamics. Now the key issue is how to preserve the properties that characterize the original network in the downscaled network so that accurate results can be extrapolated after the packet-level simulation. In particular, the network property of interest should remain invariant in the process of scaling down and rescaling up the network.

For the purpose of simulating large-scale TCP/IP networks, we use the *bandwidth-delay* product as the network property to be preserved, as it represents the capacity of the "pipe," i.e., the amount of data packets that can be transmitted without waiting for the acknowledgment [18]. That is, we scale down the original network by reducing the link capacity by a fraction α ($0 < \alpha \leq 1$), increasing the link delay by $\frac{1}{\alpha}$, but keeping the bandwidth-delay product constant. (Note that we change neither the number of nodes/flows nor the queue (the maximum buffer size or the AQM parameters.)) By preserving this product invariant during the down/up scaling operations, the network capacity as perceived by each TCP connection is preserved, and we can formally prove that the queue dynamics (e.g., the instantaneous queue length), the RTT dynamics, and the TCP window size dynamics remain unchanged in the operations. We have also carried out *ns-2* simulation comparing RSM based simulation against packet level simulation, with respect to the capability of capturing the transient, packet-level network dynamics, the execution time, and the discrepancy in simulation results. The simulation results indicate an order of magnitude or more improvement (maximally 50 times) in execution time and the performance improvement becomes more prominent as the network size increases (in terms of number of nodes and network capacity) or as the scaling parameter decreases. The error discrepancy between RSM based simulation and packet level simulation, on the contrary, is minimally 1-2 % and maximally 10 % in a wide variety of network topologies (with various AQM strategies) and traffic loads.

The notion of scaling down/up the network to facilitate network monitoring and performance prediction has also been used in Small-scale Hi-fidelity Reproduction of Network Kinetics (SHRiNK) [17]. SHRiNK proposes two methods: one deals with IP networks with long-lived flows and the other with a mixture of long-lived and short-lived flows. Both methods reduce the amount of data packets by sampling real traffic over the network, and then simulate in a downscaled network with the sampled traffic in order to predict the behavior in the original network. In the first method, the link capacity, the maximum buffer size (along with AQM parameters, e.g., the minimum/maximum thresholds in RED, used) at each link, and the number of flows are proportionally reduced, while the end-to-end delay is kept as the invariant. (As will be elaborated on in Section 2, the queue dynamics is changed under the first method.) The second method resembles RSM in that the link capacity is reduced and the link delay is proportionally increased. Their theoretical analysis and simulation study to validate their design, however, focus only on the first method, and the theoretical base for the second method is simply lacking. Moreover, in order not to change the property of the original input traffic, both methods rely heavily on the assumption that each input flow is a Poisson process.

The rest of the paper is organized as follows. In Section 2, we give a summary of existing work that aim to expedite network simulation. In Sections 3–4, we present RSM, formally prove that it preserves network dynamics, and validate its correctness. Following that we elaborate on how we implement RSM in *ns-2*, and present our simulation results in Section 5. Finally we conclude the paper in Section 6.

2. RELATED WORK

In this section we summarize existing work that pertains to the issue of expediting network simulation, while retaining its accuracy.

Fluid model-based simulation:
Several research efforts have focused on fluid model based simulation. Liu *et al.* [11] demonstrated the fundamental performance gain in fluid model based simulation over, rather than a realistic network with detailed network protocols, simple network components. Milidrag *et al.* [13] presented various sets of differential equations that describe the behaviors of network components in the continuous time domain. They showed that as long as the behavioral characteristics in the continuous time domain can be exactly specified, fluid simulation gives results with reasonable error bounds. Wu *et al.* [21] studied the error behavior that simulation results exhibit in a simple $M/D/1$ network configuration.

Fluid models have also been used to study the throughput behavior of TCP and congestion control algorithms, together with active queue management in the steady state [15, 16, 19]. Liu *et.al.* [12] solve one of the existing fluid models with the numerical Runge-Kutta method, and incorporate numeric results in the simulation of large scale IP networks. Kim and Hou [8] investigate the feasibility of fluid model based simulation for IEEE 802.11-Operated wireless LANs (WLANs), in which a throughput model is developed to describe data transmission activities in wireless LANs and then used to implement fluid model based simulation in *ns-2*. Their results show two orders of a magnitude improvement with acceptable error bounds. As indicated in Section 1, fluid model based simulation may not render satisfactory performance (in terms of the discrepancy between results obtained in packet level simulation and fluid model based simulation) in the case of light and/or sporadic traffic, as it relies on the assumption of existence of a large number of flows [12, 15, 19]. It is also limited to show kinetic transient behaviors of the network.

Network calculus-based simulation:

Kim and Hou [9] examine the feasibility of incorporating network calculus models in simulating TCP/IP networks. By exploiting network calculus properties, they characterize how TCP congestion control — additive increase and multiplicative decrease (AIMD) — interact with AQM strategies in the analytic model, and regulate TCP flows in a simulation engine with the derived model. They show that as compared to time stepped hybrid fluid simulation (TSHS), significant improvement can be made in expediting the simulation, while keeping the error discrepancy reasonably small. As indicated in Section 1, although network calculus-based simulation gives accurate steady state system throughput, it cannot show, due to the nature of network calculus, the transient behavior of the network, e.g., the instantaneous queue length and the packet dropping probability at each bottleneck link.

Simulation based on scaling down the network: Pan *et al.* [17] introduce two SHRiNK methods to sample, simulate, and predict the network behavior. The first method deals with IP networks with long-lived flows and the other with a mixture of long-lived and short-lived flows. A proportion of the traffic flows are independently sampled and collected. The original network is reduced to a downscaled one and fed with the sampled traffic. In the first method, the downscaling operation is performed by reducing the number of flows, the link capacity, the maximum buffer sizes, and AQM parameters, while keeping the end-to-end delay invariant. In order to keep the end-to-end delay invariant, the queue dynamics has to be approximated (by certain linear functions) and may respond differently to each TCP connection. As a result, the network capacity as perceived by each TCP connection during its interaction with the network is changed. Moreover, in cases where the queue dynamics cannot be adequately approximated with linear functions, e.g., when DropTail is used as the AQM strategy, SHRiNK cannot operate correctly. Although similar to RSM, the second SHRiNK method was presented without theoretical reasoning. In addition, both SHRiNK methods rely heavily on the assumption that each input flow is a Poisson process. RSM can be universally applied to long-lived and shorted-lived TCP connections, the down/up scaling operations in RSM do not change the queue dynamics (which will be rigorously proved), and its correctness does not rely on any assumption.

3. RESCALING SIMULATION METHODOLOGY

In this section, we present the rescaling simulation methodology. The key idea of the methodology is to preserve the network capacity (as determined by the queue dynamics) during the down/up scaling operation. We first discuss the network invariant in simulating TCP/IP networks. Then we introduce the rescaling simulation model based on the network invariant.

3.1 Network Invariant

Our objective is to scale down a large scale network to a small one that can be simulated in a short period of time to produce sufficient results. The down scaling operation expedites network simulation by reducing the number of events generated/processed in the simulation. For example, if we reduce the link capacity at each link, we can reduce the number of sending and receiving events per unit time at the link. Similarly, if we increase the propagation delay at each link, we can reduce the sending rate of each TCP connection (as a result of increased round trip time.)

One important issue is then how to scale down the network so that the simulation results obtained from the downscaled network can be used to accurately infer those corresponding to the original network. We claim that the down scaling operation should not change the network capacity as perceived by TCP connections. Note that the network capacity is determined by the network dynamics (such as the instantaneous queue length, to which TCP connections respond) and is reflected upon the TCP throughput. Hence for the purpose of simulating large-scale TCP/IP networks, we use the *bandwidth-delay* product as the network invariant to be preserved during the down/up scaling operations. In the perspective of a TCP connection, the bandwidth-delay product represents the amount of data packets that can be in transit without waiting for the acknowledgment [18].

Specifically, let B and D denote, respectively, the available bandwidth and delay along a path in the original network, and P_{BDP} the bandwidth-delay product along the path. For notational convenience, we denote the corresponding variables in the downscaled network by attaching an apostrophe to the variables, i.e., B', D', and P'_{BDP}. The following constraint is used in the down/up scaling operations:

$$P_{BDP} = B \cdot D = B' \cdot D' = P'_{BDP}. \quad (1)$$

By Eq. (1), a scaling parameter, α, is determined as follows:

$$B' = \alpha \cdot B, \quad (2)$$

$$D' = \frac{D}{\alpha}, \quad (3)$$

where $0 < \alpha \leq 1$.

Note that we do not change the number of nodes, the number of flows, or the queue-related parameters (e.g., the maximum buffer size or the AQM parameters). The only parameters that are scaled are the link capacity and the link delay. As will be formally proved in Section 3.2, by keeping P_{BDP} invariant, the queue dynamics (e.g., the instantaneous queue length), the RTT dynamics, and the window size dynamics remain unchanged in the operations, and hence the network capacity as perceived by each TCP connection is preserved. As a result, the downscaled network exhibits exactly the same behavior as the original network.

3.2 Rescaling Simulation Model

We now propose the rescaling simulation model. Let N denote the number of flows sharing a path with the bottleneck link of capacity C. Let $q(t)$ and $p(t)$ denote, respectively, the queue length and the packet dropping probability of the bottleneck link at time t, and T the propagation delay of the path. Let $W_i(t)$ and $R_i(t)$ denote, respectively, the window size and the round trip time of flow i at time t. Then the interaction between TCP flows and the bottleneck link can be characterized with the TCP model given in [14]:

$$R_i(t) = T + \frac{q(t)}{C} \quad (4)$$

$$\frac{dq}{dt} = \sum_{i=1}^{N} \frac{W_i(t)}{R_i(\tau_i)} - C \quad (5)$$

$$\frac{dW_i(t)}{dt} = \frac{1}{R_i(t)} - \beta \cdot W_i(t) \cdot \frac{W_i(\tau_i)}{R_i(t)} \cdot p(\tau_i), \quad (6)$$

where β is the multiplicative parameter, $p(\cdot)$ is the packet dropping probability, and $\tau_i = t_i - R_i(t)$.

The rescaling simulation model reduces the link bandwidth and increases the link delay by the scaling parameter, α (Eqs. (2) and (3)). Hence, the time instants at which packet events occur are also delayed by $\frac{1}{\alpha}$, since each packet is served at α times smaller bandwidth (which stretches its transmission time by $\frac{1}{\alpha}$) and experiences $\frac{1}{\alpha}$ times larger delay. Specifically, a packet event that occurs at time t in the original network is now delayed to t' in the downscaled network as follows:

$$t' = \frac{1}{\alpha} \cdot t, \tag{7}$$

where $t(t \geq 0)$ and $t'(t' \geq 0)$ define, respectively, a time instant for the original network and for the downscaled network.

In what follows, we will investigate the dynamics of the queue length, the round trip time, and the TCP window size in both the downscaled network and the original network. Again we denote the corresponding variables in the downscaled network by attaching an apostrophe to the variables.

3.2.1 Queue Dynamics

We will show that queue length of the bottleneck link in the original network is the same to that in the downscaled network:

$$q'(t') = q(t) \quad \text{and} \quad \dot{q'} = \dot{q}$$

This implies that the downscaled network responds to each TCP connection in exactly the same manner as the original network does. Moreover, this is achieved without modifying or approximating any AQM parameter in the downscaled network.

We first look at $\frac{dq'}{dt}$:

$$\begin{aligned}
\frac{dq'}{dt} &= \sum_{i=1}^{N} \frac{P'_{i,BDP}(t)}{D'_i(t)} - C' \\
&= \sum_{i=1}^{N} \frac{P_{i,BDP}(t)}{\frac{D_i(t)}{\alpha}} - \alpha \cdot C \\
&= \alpha \cdot \left\{ \sum_{i=1}^{N} \frac{P_{i,BDP}(t)}{D_i(t)} - C \right\} \\
&= \alpha \cdot \frac{dq}{dt} \\
&= \frac{dq}{\frac{1}{\alpha} \cdot dt}.
\end{aligned} \tag{8}$$

Note that in Eq. (8), the change in the queue size is simply the difference between the arrival rate of all flows and the link capacity, and the arrival rate of a flow i is obtained by dividing its current bandwidth-delay product ($P_{i,BDP}(t)$) by its current delay ($D_i(t)$).

Since $t' = \frac{1}{\alpha} \cdot t$ (Eq. (7)), $dt' = \frac{1}{\alpha} \cdot dt$, and thus,

$$\frac{dq'}{dt} = \frac{dq}{dt'},$$

or

$$dq' \cdot dt' = dq \cdot dt. \tag{9}$$

Since $q'(0) = q(0) = 0$ and Eq. (9) is a separable first order equation [4], we can obtain the following result by integrating both sides of Eq. (9) when $t, t' \geq 0$:

$$q'(t') = q(t). \tag{10}$$

By Eq. (10), we know as long as the down scaling operation preserves the *bandwidth-delay* product of the original network, the queue dynamics in both the original network and the downscaled network are the same at each event time.

3.2.2 RTT Dynamics

We can compute the round trip time in the downscaled time domain, t', as follows. Since $R_i(t) = T + \frac{q(t)}{C}$,

$$\begin{aligned}
R'_i(t') &= T' + \frac{q'(t')}{C'} \\
&= \frac{T}{\alpha} + \frac{q(t)}{\alpha \cdot C} \\
&= \frac{1}{\alpha} \cdot (T + \frac{q(t)}{C}) \\
&= \frac{1}{\alpha} R_i(t).
\end{aligned} \tag{11}$$

Note that we have used $q'(t') = q(t)$ in the second equality in Eq. (11).

With Eq. (11), we can now re-express $\frac{dq'(t')}{dt}$ in Eq. (8) as follows:

$$\begin{aligned}
\frac{dq'}{dt} &= \sum_{i=1}^{N} \frac{W'_i(t)}{R'_i(\tau'_i)} - C' \\
&= \sum_{i=1}^{N} \frac{W_i(t)}{\frac{R_i(\tau_i)}{\alpha}} - \alpha \cdot C \\
&= \alpha \cdot \left\{ \sum_{i=1}^{N} \frac{W_i(t)}{R_i(\tau_i)} - C \right\} \\
&= \alpha \cdot \frac{dq}{dt},
\end{aligned}$$

where the first equality results from the fact that the current TCP window size ($W_i(t)$) can be represented by the current bandwidth-delay product ($P_{i,BDP}(t)$) as perceived by each TCP connection i.

3.2.3 Window Size Dynamics

Eq. (11) implies that each TCP connection in the downscaled network exhibits the same window dynamics but the response is $\frac{1}{\alpha}$ times slower than that in the original network (since a TCP connection in the downscaled network adjusts its rate per round trip time $R'_i(t)$). To further verify this, we define the window dynamics in the downscaled network based on Eq. (6) as follows:

$$\frac{dW'_i}{dt'} = \frac{1}{R'_i(t')} - \beta \cdot W'_i(t') \cdot \frac{W'_i(\tau'_i)}{R'_i(t')} \cdot p(\tau'_i), \tag{12}$$

where $\tau'_i = t' - R'_i(t')$.

By plugging Eq. (11) into Eq. (12), we have

$$\begin{aligned}
\frac{dW'_i}{dt'} &= \frac{1}{\frac{1}{\alpha} \cdot R_i(t)} - \beta \cdot W_i(t) \cdot \frac{W_i(\tau_i)}{\frac{1}{\alpha} \cdot R_i(t)} p(\tau_i) \\
&= \alpha \cdot \left\{ \frac{1}{R_i(t)} - \beta \cdot W_i(t) \cdot \frac{W_i(\tau_i)}{R_i(t)} \cdot p(\tau_i) \right\} \\
&= \alpha \cdot \frac{dW_i}{dt},
\end{aligned} \tag{13}$$

where $\tau_i = t - R_i(t)$. Note that in the first equality of Eq. (13), we use the current bandwidth-delay product of the path as the current window size, and hence $W_i'(x') = W_i(x)$, for $\forall x' = \frac{1}{\alpha} \cdot x$. Additionally, $\tau_i' = \frac{1}{\alpha} \cdot \tau_i$ since $R_i'(t') = \frac{1}{\alpha} \cdot R_i(t)$ (Eq. (11)), and $p(\cdot) = p'(\cdot)$ since $q'(t') = q(t)$ (Eq. (10)), for $\forall t' = \frac{1}{\alpha} \cdot t$.

As implied in Eqs. (11) and (13), all the events that occur in the original network are delayed by the factor of $\frac{1}{\alpha}$ in the downscaled network. This is consistent with Eq. (7).

3.2.4 Comparison with SHRiNK

As compared to SHRiNK [17], RSM possesses several desirable features: (i) RSM does not make any assumption on the input traffic, while SHRiNK relies heavily on the Poisson assumption for the input traffic. (ii) RSM preserves the queue dynamics and hence preserves the network capacity as perceived by TCP connections. In contrast SHRiNK approximates the queue dynamics in the downscaled network in order to preserve the queuing delay. As a result, the downscaled network may respond differently to TCP connections. (iii) RSM can work together with any AQM strategy, while SHRiNK cannot be used if the modified queue dynamics cannot keep the queuing delay invariant in the downscaled network. For example, as reported in [17], if the AQM strategy is DropTail, SHRiNK cannot produce accurate results.

4. VALIDATION OF RSM WITH MATLAB

In this section, we validate RSM (Section 3) with *MATLAB* [20]. The specific effect when we use the RSM model with a scaling parameter α is that all the events that occur at time $t(t \geq 0)$ in the original network are delayed to $t'(t' \geq 0) = \frac{1}{\alpha} \cdot t$ in the downscaled network. This is attributed to the fact that the link bandwidth is scaled down by the factor of α and the link delay is scaled up by the factor of $\frac{1}{\alpha}$ during the down-scaling operation. The dynamic network behavior that the original network exhibits at any time instant can be observed in the downscaled network, except that the time instant at which each event occurs is delayed by the factor $\frac{1}{\alpha}$ (Eqs. (13) and (7)). In what follows we investigate the trajectory of the queue length and transient alteration of the dropping probability at a link to verify Eqs. (13) and (7).

Figure 1 gives a *MATLAB Simulink* diagram that implements the interaction between a bottleneck link (equipped with RED) and multiple TCP flows. In the figure, the *RTT module* executes the operation given in Eq. (4), the *queue module* executes the function given in Eq. (5), the *RED module* performs the RED operations at the link [6] and the *TCP module* carries out the dynamics given in Eq. (6). Additionally, the *prop. delay* module and the *link capacity* module represent, respectively, the delay and the capacity of the link, and N is the number of flows.

Figures 2 and 3 give, respectively, the kinetic dropping probability and the instantaneous queue length for the bottleneck link in Figure 1 ($N = 100$). Figure 2 (a) presents the dynamic trajectories of the packet dropping probability of the link in the down-scaled network, where each trajectory corresponds to a different value of the scaling parameter α. Note that events are delayed by the factor of $\frac{1}{\alpha}$. Figure 2 (b) gives the trajectories of the packet dropping probability after the network is re-scaled up with α. Note that all the trajectories agree with one another. Figure 3 gives the instantaneous queue length of the link in the down-scaled network ((a)) and after the network is re-scaled up ((b)). Conclusions similar to those made for Figure 2 can be drawn.

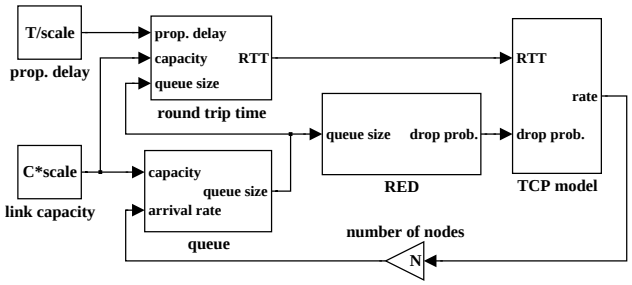


Figure 1: The MATLAB Simulink diagram for TCP dynamics

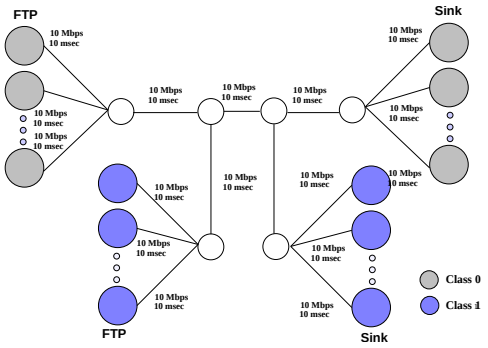


Figure 4: The network configuration used in the second set of experiments.

5. SIMULATION STUDY

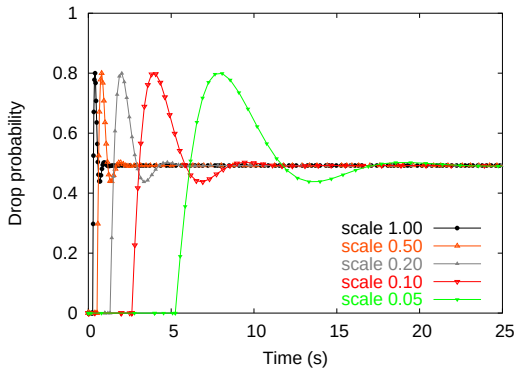
In this section, we discuss how we implement RSM in *ns-2* [2] and conduct a simulation study, comparing the network behaviors (e.g., the queue dynamics) and the steady state behavior (e.g., the TCP throughput) obtained in RSM-based simulation and packet level simulation.

5.1 Implementation

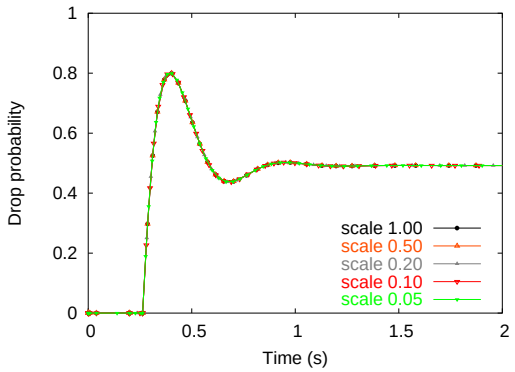
Recall that the key idea of RSM is to reduce the number of events in the simulation by scaling down the original network (i.e., decreasing the link capacity and increasing the link delay by the same factor), while preserving the bandwidth-delay product invariant in the down-scaled network (Section 3). As a result, neither the number of flows nor the queue-related parameters (e.g., the maximum buffer size and AQM parameters) need to be changed. As a matter of fact, the implementation of RSM is quite simple and straightforward. The network simulator takes as input the network topology and the scaling parameter, scales down the original network, carry out packet-level simulation in the down-scaled network for a short period of time, and then extrapolates the results corresponding to the original network. Only a light-weight preprocessor and a post-processor are needed to scale down the network and to extrapolate the simulation results.

5.2 Simulation

We have carried out an extensive *ns-2.1b9a* simulation study in a wide variety of network topologies and traffic loads, to assess the effectiveness of RSM in terms of reducing the execution time and capturing both the transient, packet-level network dynamics and the steady state network performance. Due to the space limit, in what follows we present

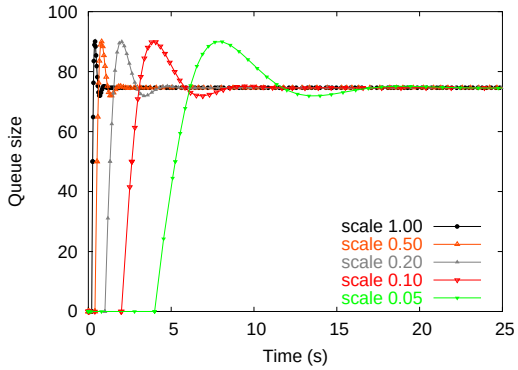


(a) Packet dropping probability in the down-scaled network

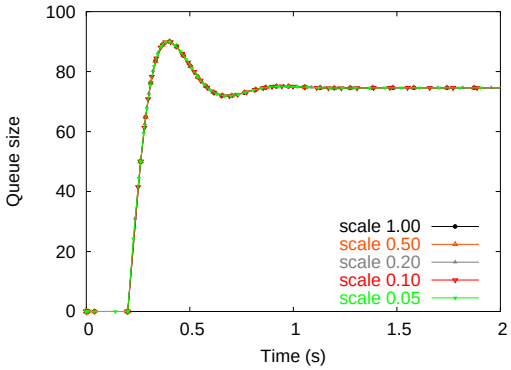


(b) Packet dropping probability after the network is re-scaled up

Figure 2: Trajectories of the packet dropping probability in the down-scaled network and after the network is re-scaled up with the same scaling parameter α.



(a) Queue length in the down-scaled network



(b) Queue length after the network is re-scaled up.

Figure 3: Trajectories of the instantaneous queue length in the down-scaled network and after the network is re-scaled up with the same scaling parameter α.

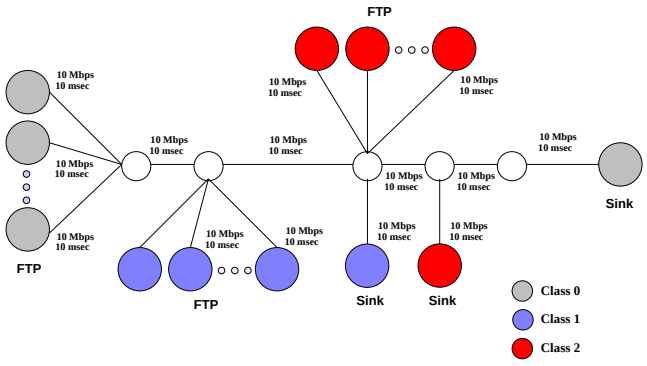


Figure 5: The network configuration used in the third set of experiments.

only simulation results obtained in the two network configurations given in Figures 4 and 5. (Note that the configuration in Figure 4 has one bottleneck, while that Figure 5 has two bottleneck links.) Both the link bandwidth and the link delay are labeled in the figure, unless otherwise specified. Both short-lived and long-lived TCP connections, as well as different variations of TCP, are used in the transport layer, but due to the space limit, we only present results with *TCP Reno* connections.

Each router is equipped with a buffer of size 100 packets, and the default packet size is set to be 500 bytes. Different AQM strategies are employed at routers in different simulation runs. The setting of parameters in the various AQM strategies is as follows. The minimum and maximum threshold of RED is 30 and 70, respectively. The update interval, reference value, and gain of REM are 10 ms, 50, and 0.1, respectively. The reference queue size and sampling frequency of PI is 50 and 100 times per second, and the remaining parameters, such as kp and ki (which in turn decides a and b) are determined in compliance with [7]. The desirable utilization, γ, of AVQ is set to 0.98 , and the damping factor, α, is determined in compliance with Theorem 1 in [10] to ensure system stability ($\alpha = 0.15$).

All the experiments are conducted in Linux 2.4.18 on a Pentium 4/1.9 *Ghz* PC with 1 *GBytes* memory and with 2 *GBytes* swap memory.

5.2.1 Performance in the presence of long-lived TCP connections

In this section, we study how effective RSM based simulation is in simulating long-lived TCP flows.

Performance of RSM based simulation w.r.t. network dynamics

First, we examine how closely the network dynamics obtained under RSM based simulation exhibit to that under packet level simulation. Figure 6 depicts the instantaneous queue length versus time in the network configuration given in Figure 4. The number of nodes in each class varies from 5 to 100, and the scaling parameter varies from 0.02 to 1.0 in these simulation runs. Also, router nodes employ a different AQM strategy in each simulation run. Due to the space limit, we only present in Figure 6 the case of 100 nodes per class. Regardless of the AQM strategy employed or the value of the scaling parameter, the queue length extrapolated from the downscaled network agrees extremely well with that observed in the original network.

Figure 7 depicts the instantaneous queue length of the second bottleneck link versus time in the network configuration in Figure 5, but two cases are presented: the case of 20 nodes per class and the case of 100 nodes per class. Again the instantaneous queue lengths extrapolated from the downscaled network (the curves labeled with "scale 0.2") agree extremely well with that in the original network.

Performance of RSM based simulation w.r.t. error discrepancy

We now quantitatively evaluate the discrepancy between results obtained in RSM based simulation and those in packet level simulation. In both simulation modes, the TCP throughput is measured at a receiver and summed up to give the total throughput per class and the total throughput for the network. Figure 8 gives the total number of packets received at class 0 nodes and at all the nodes, in the network configuration given in Figure 4. Each simulation run lasts for 1000 seconds. The error discrepancy observed between two simulation modes is at most approximately 10 % of the capacity of the bottleneck link.

Figure 9 gives the total number of packets received at class 0 nodes and at all the nodes, in the the network configuration given in Figure 5. Again each simulation run lasts for 1000 seconds. Again the error discrepancy is at most approximately 10 % of the capacity of the bottleneck link.

Performance of RSM based simulation w.r.t. execution Time

We now evaluate the performance gain of RSM (as compared to packet level simulation) in terms of the execution time required to carry out the simulation. Figure 10 depicts the execution time versus the number of nodes in a 1000-second simulation run in the network configuration given in Figure 4. The simulation results indicate an order of magnitude or more improvement (maximally 50 times) in execution time and the performance improvement becomes more prominent as the network size increases (in terms of number of nodes and network capacity) or as the scaling parameter decreases.

Figure 11 depicts the execution time versus the number of nodes in a 1000-second simulation run in the network configuration given in Figure 5. The speed-up in execution time as a result of using RSM can be as large as approximately 50 times.

5.2.2 Performance in the presence of long- and short-lived TCP connections

In this section, we explore more dynamic scenarios in which long-lived and short-lived TCP connections co-exist and interfere with each other. Each short-lived connection is generated by a Pareto traffic generator in *ns-2* in the way that packets are sent at a fixed rate of 200 Kbps during the on periods and no packets are sent during the off periods. The length of each on/off period follows a Pareto distribution with the Pareto shape parameter of 1.5 and the mean value of 100 *ms*. The application frame size is set 210 bytes. Each simulation run lasts for 300 seconds, and short-lived connections are only active in the interval [100, 200] seconds. (We have also carried out experiments in which all the TCP connections are short-lived. As the results exhibit similar trends, they are not repeat here.)

Performance of RSM based simulation w.r.t. network dynamics

Figure 12 depicts the instantaneous queue length versus time in the network configuration given in Figure 4, except that the capacities of all the link, excluding the bottleneck link, are increased to 100 Mb. There are 100 nodes per class. As shown in Figure 12, the dynamic changes of the queue length in the downscaled network agree very well with the original behavior.

It should be noted that the simulation results in Figure 6 are the instantaneous queue lengths rescaled from the downscaled network, while those in Figure 12 are not rescaled. The same queue dynamics can still be observed even if the events (arrival/departure of short-lived TCP connections) are not delayed (by the factor of α) in the time-stretched, down-scaled network. This result shows that we do not have to worry about one pitfall of RSM-based simulation that if the simulation time is not proportionally increased in simulating the downscaled network, some of the dynamic events may not be executed in time before the simulation ends. Figure 13 gives the total number of packets received at class 0 nodes in the network configuration given in Figure 4. Error discrepancy claims that similar to those in the presence of only long-lived TCP connections can be make here.

Performance of RSM based simulation w.r.t. error discrepancy

We again quantitatively evaluate the discrepancy between results obtained in RSM based simulation and those in packet level simulation. Figure 13 gives the total number of packets received at class 0 nodes in the network configuration given in Figure 4. Each simulation run lasts for 300-seconds. The error discrepancy observed between two simulation modes is still within approximately 10 % of the capacity of the bottleneck link.

Performance of RSM based simulation w.r.t. Execution Time

We evaluate again the performance gain of RSM (as compared to packet level simulation) in terms of the execution time required to carry out the simulation. Figure 14 depicts the execution time versus the number of nodes in a 300-second simulation run in the network configuration given in

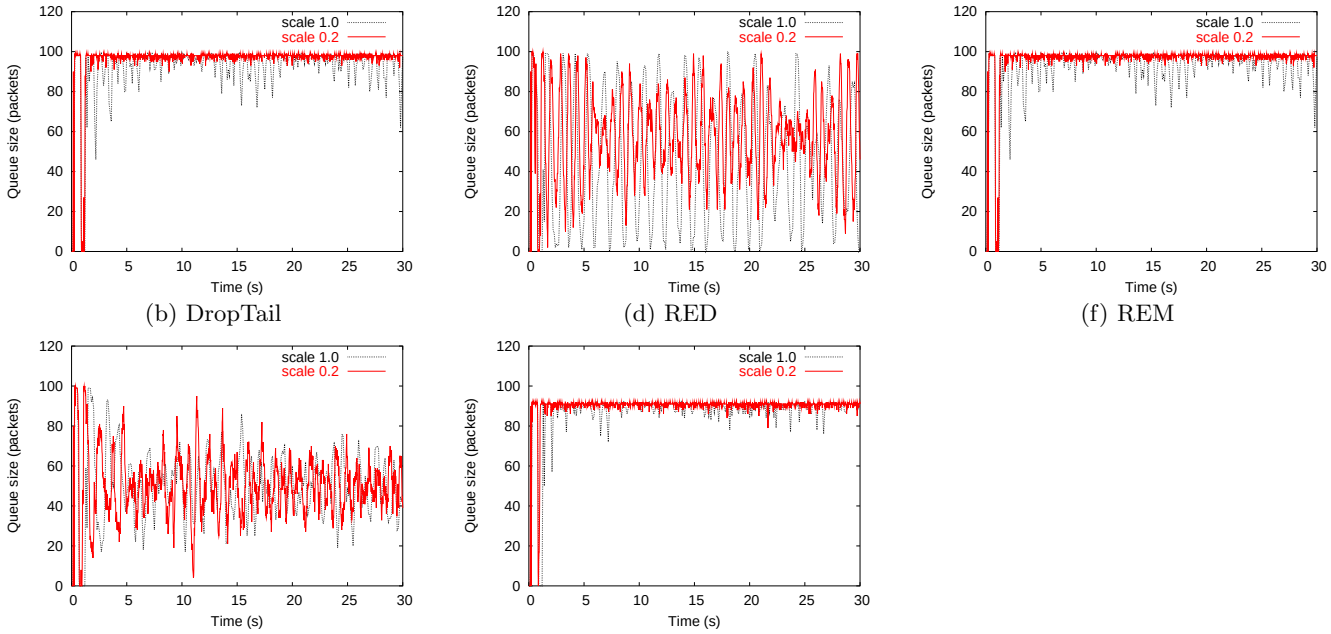


Figure 6: Instantaneous queue length versus time in the network configuration given in Figure 4 when the number of nodes per class is 100. The curve labeled with "scale 1.0" is the instantaneous queue length in the original network and that labeled with "scale 0.2" is the instantaneous queue length extrapolated from the down scaled network.

Figure 4. The same conclusion made in the case of only long-lived TCP connections can be applied here, except that a performance improvement of at most 30 times (rather than 50 times) is observed.

6. CONCLUSION

In this paper, we present the rescaling simulation methodology (RSM) for simulating large-scale TCP/IP networks. Specifically, we scale down the original network by reducing the link capacity by a fraction α, increasing the link delay by $\frac{1}{\alpha}$, but keeping the bandwidth-delay product constant. (Note that we change neither the number of nodes/flows nor the queue (the maximum buffer size or the AQM parameters.)) By preserving this product invariant during the down/up scaling operations, the network capacity as perceived by each TCP connection is preserved, and we formally prove that the queue dynamics (e.g., the instantaneous queue length), the RTT dynamics, and the window size dynamics remain unchanged in the operations. Implementation of RSM in a network simulation is simple, straightforward, and requires only a simple preprocessor/post-processor to scale down and re-scale up the network.

We have also carried out *ns-2* simulation comparing RSM based simulation against packet level simulation, with respect to the capability of capturing transient, packet-level network dynamics, the execution time and the discrepancy in simulation results. The simulation results indicate an order of magnitude or more improvement (maximally 50 times) in execution time and the performance improvement becomes more prominent as the network size increases (in terms of number of nodes and network capacity) or as the scaling parameter decreases. The error discrepancy between RSM based simulation and packet level simulation, on the contrary, is minimally 1-2 % and maximally 10 % in a wide variety of network topologies (with various AQM strategies) and traffic loads. The encouraging simulation results, coupled with the ease in implementation, suggest that RSM can be used to simulate, and accurately infer network dynamics of, large-scale TCP/IP networks.

We have identified several directions for future work. First, we will theoretically analyze the relationship between the error discrepancy and the scaling parameter. Second, we will carry out experiments with a mixture of TCP and UDP traffic. We conjecture that UDP traffic has to be scaled down in the down scaling operation, and will look into this issue. Finally, we will also include in our study wireless LANs in part of the large scale networks.

7. REFERENCES

[1] F. Bacceli, G. Cohen, G. J. Olsder, and J.-P. Quadrat. *Synchronization and Linearity*. John Wiley & Sons, 1992.

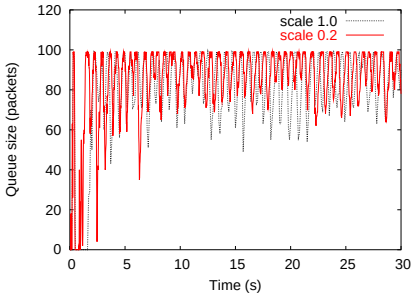
[2] U. Berkeley, LBL, USC/ISI, and X. PARC. *The ns Manual*. http://www-mash.cs.berkeley.edu/ns/, April 2002.

[3] J.-Y. L. Boudec and P. Thiran. *Network Calculus*. Springer-Verlag, 2002.

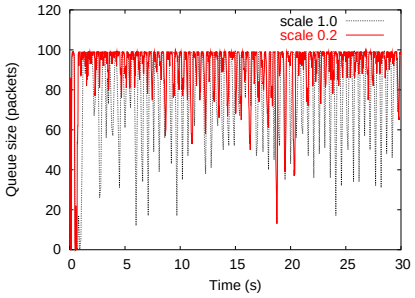
[4] C. R. Wylie Jr. *Advanced Engineering Mathematics (second edition)*. McGraw-Hill Book Company, Inc., 1960.

[5] C.-S. Chang. *Performance Guarantees in Communication Networks*. Springer-Verlag, 2000.

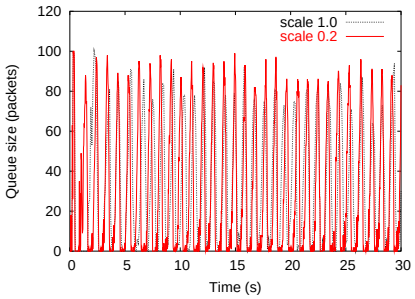
[6] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4), 1993.



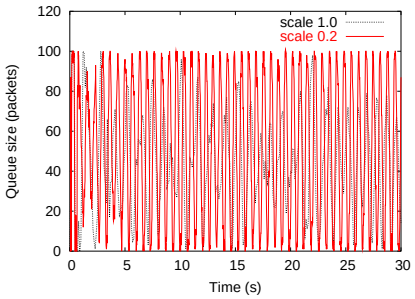
(a) DropTail with 20 nodes per class



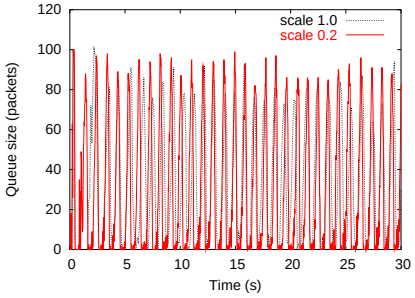
(b) DropTail with 100 nodes per class



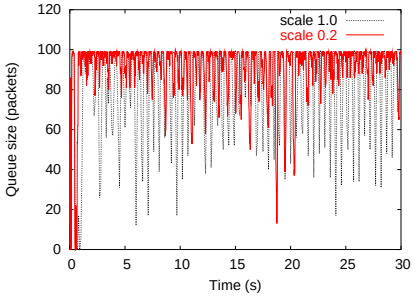
(c) RED with 20 nodes per class



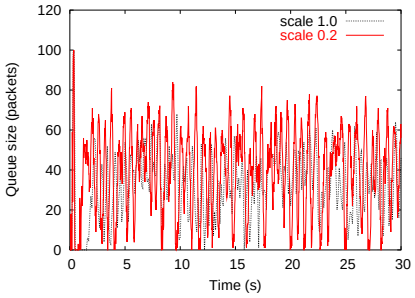
(d) RED with 100 nodes per class



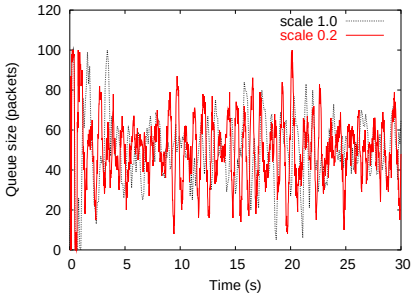
(e) REM with 20 nodes per class



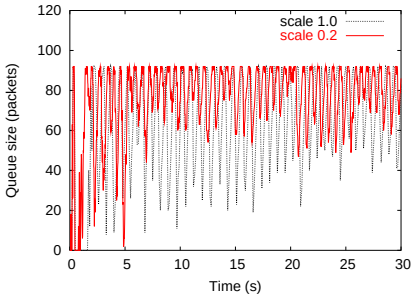
(f) REM with 100 nodes per class



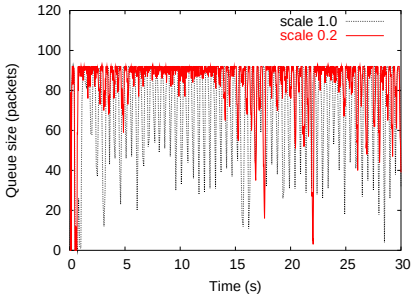
(g) PI with 20 nodes per class



(h) PI with 100 nodes per class

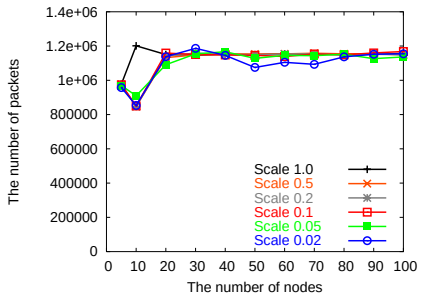


(i) Virtual Queue with 20 nodes per class

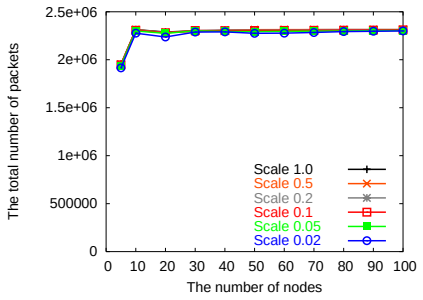


(j) Virtual Queue with 100 nodes per class

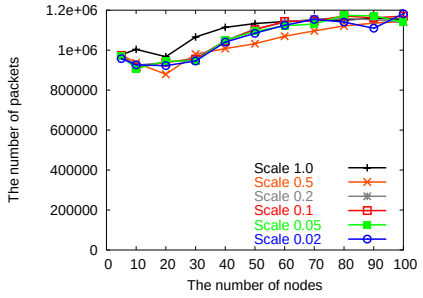
Figure 7: Instantaneous queue length versus time in the network configuration given in Figure 5.



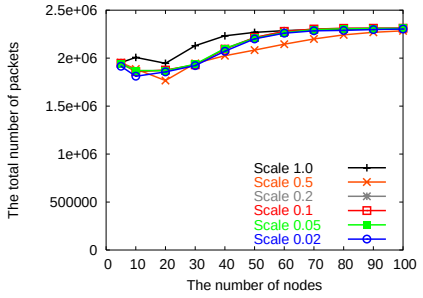
(a) Class 0 with DropTail



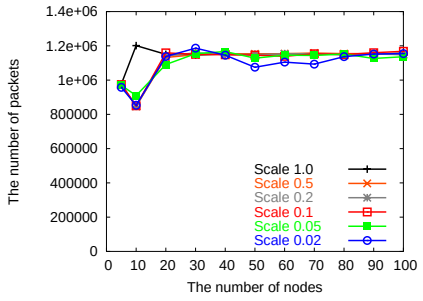
(b) All classes with DropTail



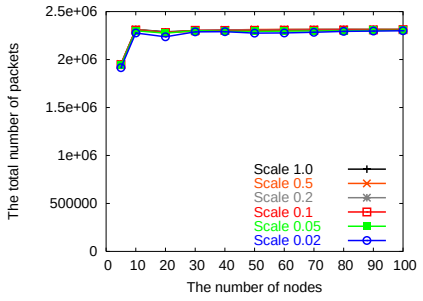
(c) Class 0 with RED



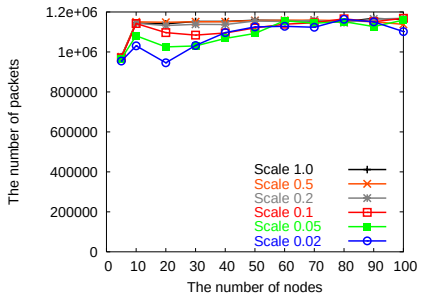
(d) All classes with RED



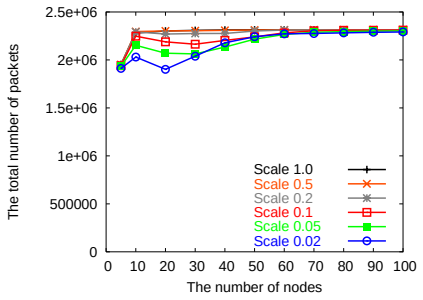
(e) Class 0 with REM



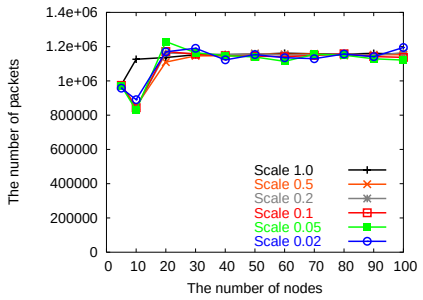
(f) All classes with REM



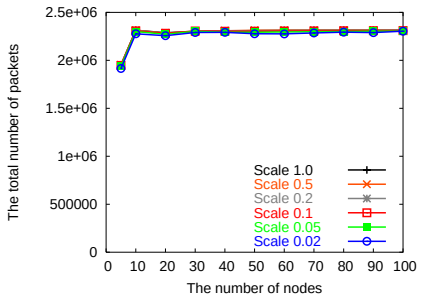
(g) Class 0 with PI



(h) All classes with PI

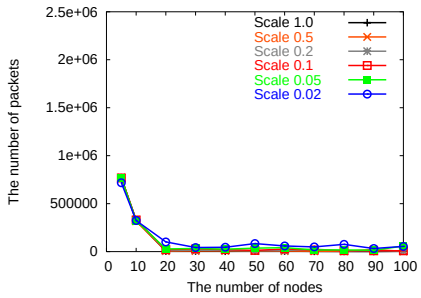


(i) Class 0 with Virtual Queue

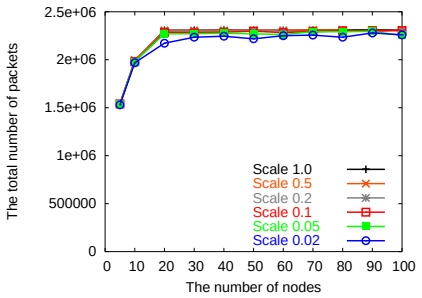


(j) All classes with Virtual Queue

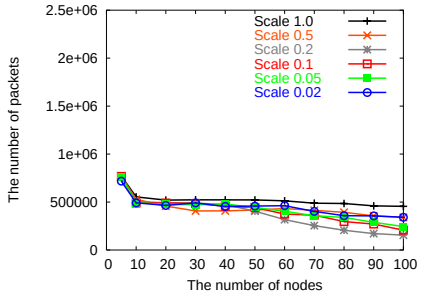
Figure 8: The total number of packets received at class 0 nodes and at all the nodes in a 1000-second simulation run in the network configuration given in Figure 4.



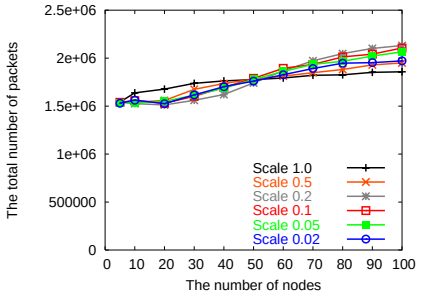
(a) Class 0 with DropTail



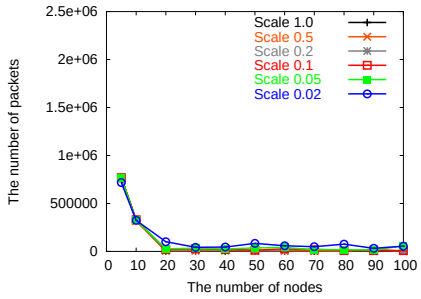
(b) All classes with DropTail



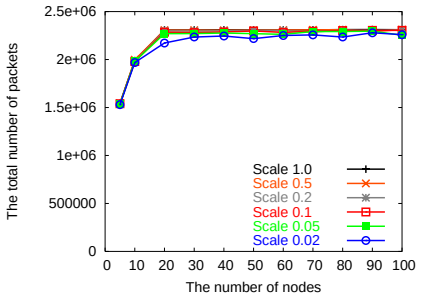
(c) Class 0 with RED



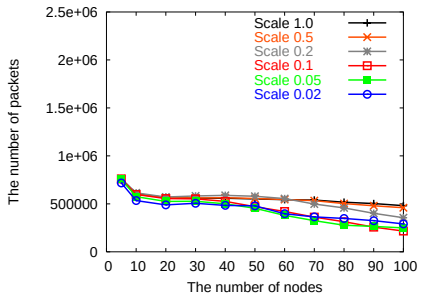
(d) All classes with RED



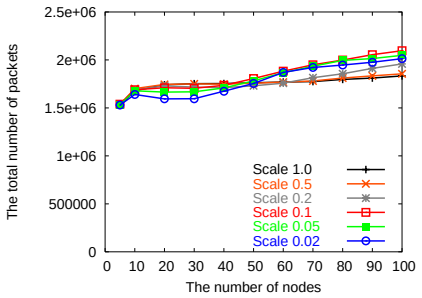
(e) Class 0 with REM



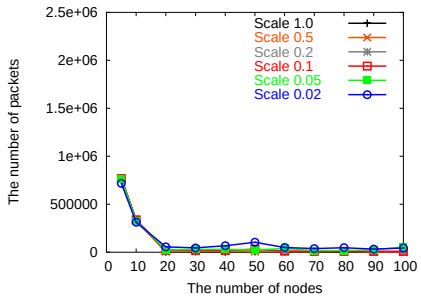
(f) All classes with REM



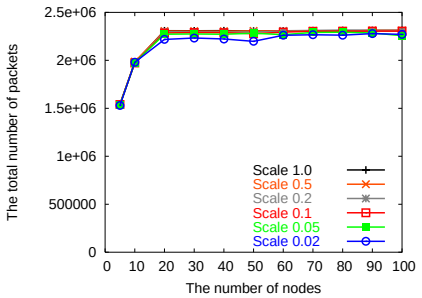
(g) Class 0 with PI



(h) All classes with PI



(i) Class 0 with Virtual Queue



(j) All classes with Virtual Queue

Figure 9: The total number of packets received at the class 0 nodes and at all the nodes in a 1000-second simulation run in the network configuration given in Figure 5.

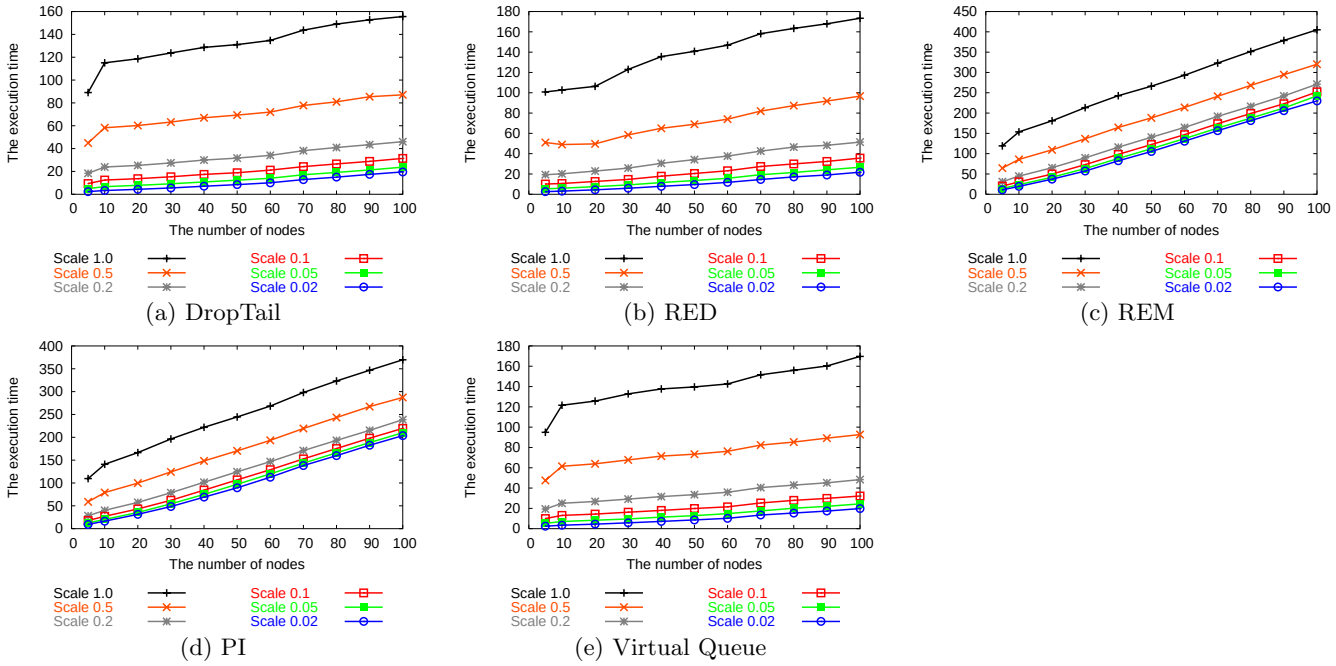


Figure 10: Execution time (sec.) required to carry out a 1000-second simulation run in the network configuration given in Figure 4.

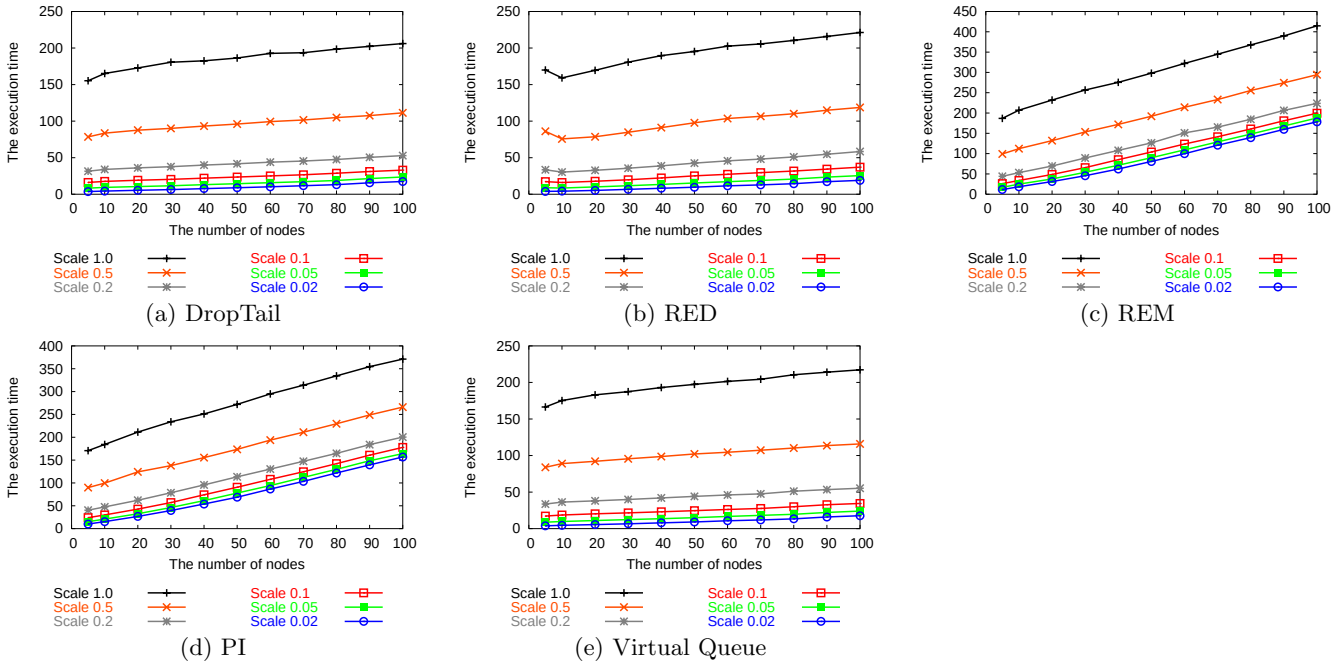
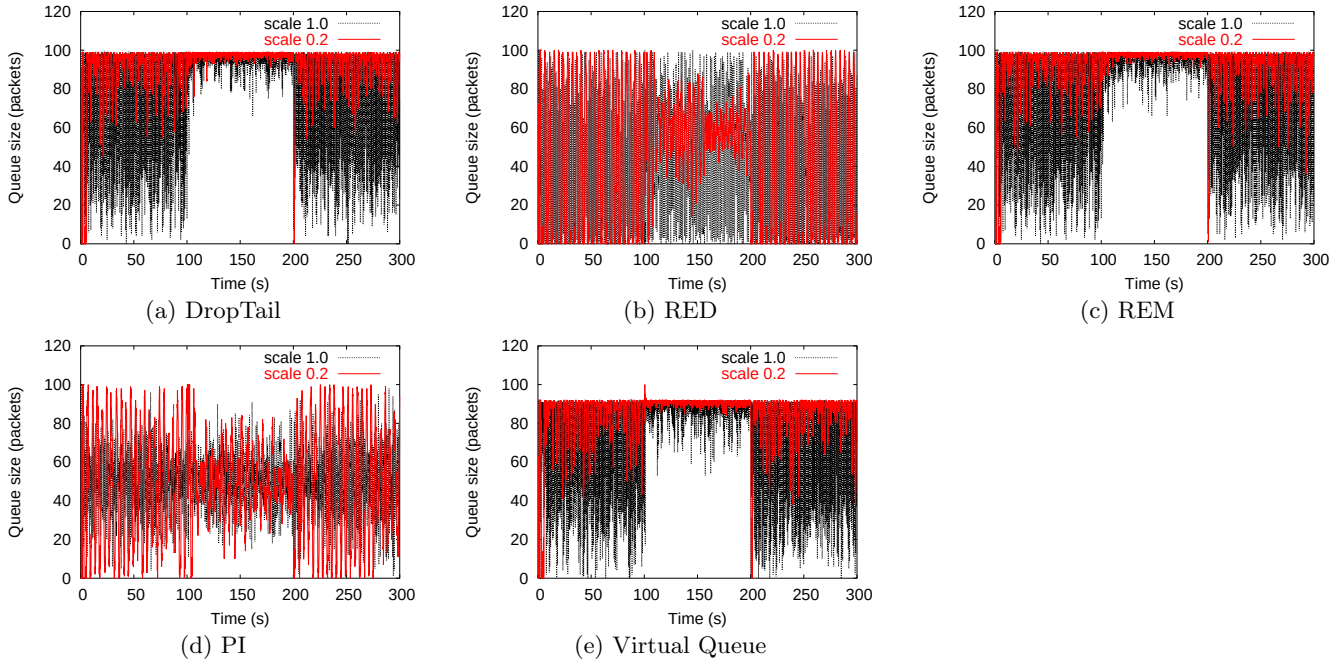


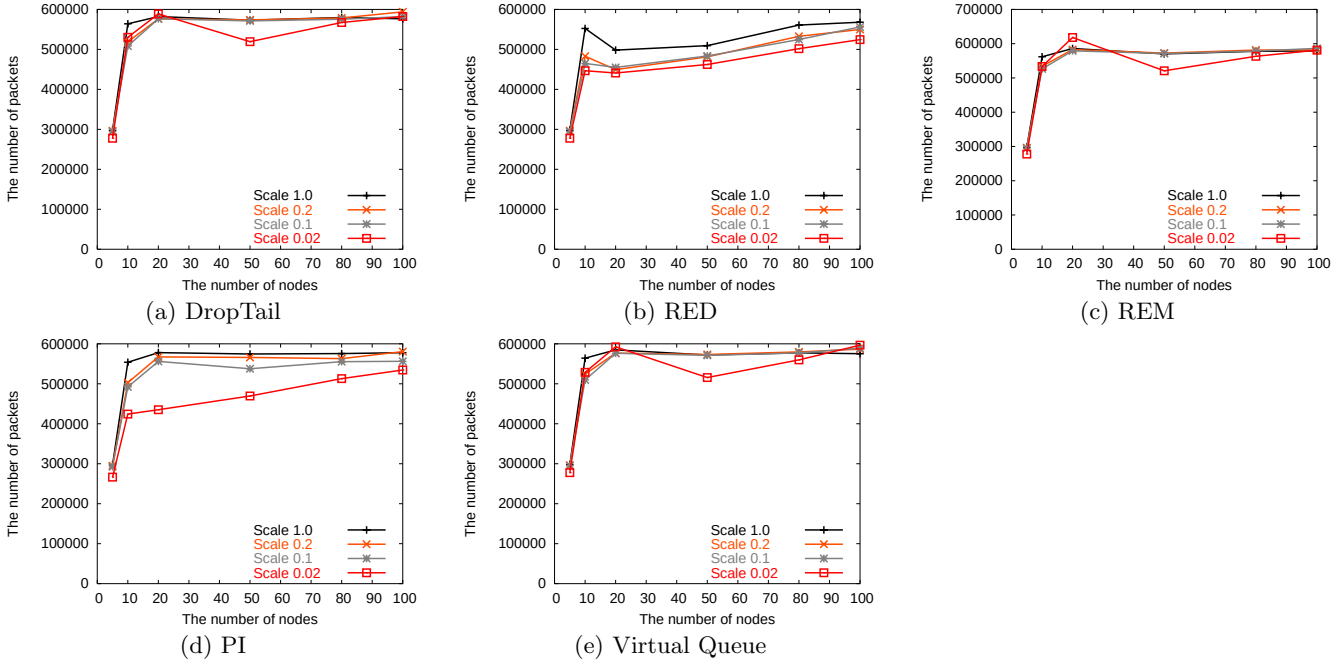
Figure 11: Execution time (sec.) required to carry out a 1000-second simulation run in the network configuration given in Figure 5.



(a) DropTail (b) RED (c) REM

(d) PI (e) Virtual Queue

Figure 12: Instantaneous queue length versus time in the presence of both long-lived and short-lived TCP connections in the network configuration given in Figure 4, except that the capacities of all the links, excluding the bottleneck link, are increased to 100 Mb. The curve labeled with "scale 1.0" is the instantaneous queue length in the original network and that labeled with "scale 0.2" is that in the downscaled network.



(a) DropTail (b) RED (c) REM

(d) PI (e) Virtual Queue

Figure 13: The total number of packets received at class 0 nodes in the presence of both long-lived and short-lived TCP connections in a 300-second simulation run in the network configuration given in Figure 4, except that the capacities of all the links, excluding the bottleneck link, are increased to 100 Mb.

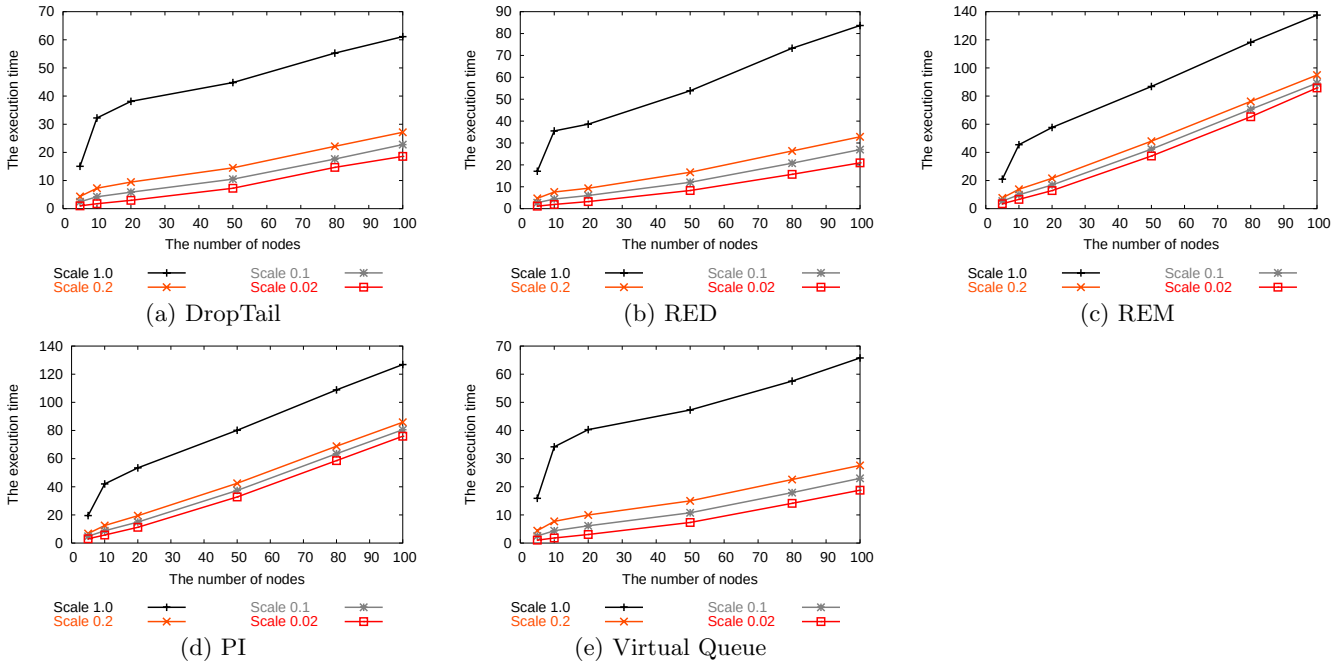


Figure 14: Execution time (sec.) required to carry out a 300-second simulation run in the presence of both short-lived and long-lived TCP connections in the network configuration given in Figure 4, except that the capacities of all the links, excluding the bottleneck link, are increased to 100 Mb.

[7] C. V. Hollot, V. Misra, D. F. Towsley, and W. Gong. On Designing Improved Controllers for AQM Routers Supporting TCP Flows. In *Proceedings of IEEE INFOCOM 2001, (Anchorage, Alaska)*, 2001.

[8] H. Kim and J. C. Hou. A Fast Simulation Framework for IEEE 802.11-Operated WLANs. In *Proceedings of ACM SIGMETRICS 2004, (New York, New York)*, June 2004.

[9] H. Kim and J. C. Hou. Network Calculus Based Simulation for TCP Congestion Control: Theorems, Implementation and Evaluation. In *Proceedings of IEEE INFOCOM 2004, (Hong Kong, China)*, March 2004.

[10] S. Kunniyur and R. Srikant. Analysis and Design of an Adaptive Virtual Queue (AVQ). In *Proceedings of ACM SIGCOMM 2001, (San Diego, California)*, August 2001.

[11] D. Liu, D. R. Figueiredo, Y. Guo, J. Kurose, and D. Towsley. Fluid models and solutions for large-scale IP networks. In *Proceedings of INFOCOM 2001, (Anchorage, Alaska)*, April 2001.

[12] Y. Liu, F. L. Presti, V. Misra, D. Towsley, and Y. Gu. Fluid Models and Solutions for Large-scale IP networks. In *Proceedings of ACM SIGMETRICS 2003 (San Diego, California)*, June 2003.

[13] N. Milidrag, G. Kesidis, and M. Devetsikiotis. An Overview of Fluid-Based Quick Simulation Techniques for Large Packet switched Communication Networks. In *Proceedings of SPIE ITCom 2001, (Denver, Colorado)*, August 2001.

[14] V. Misra, M. Gong, and D. Towsley. A Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED. In *Proceedings of ACM SIGCOMM 2000 (Stockholm, Sweden)*, September 2000.

[15] V. Misra, W. Gong, and D. Towsley. Differential Equation Modeling and Analysis of TCP Window Size Behavior. In *Proceedings of Performance 1999, (Istanbul, Turkey)*, October 1999.

[16] J. Padhye, V. Firoiu, T. Towsley, and J. Kurose. Modeling TCP throughput: A Simple Model and its Empirical Validation. In *Proceedings of ACM SIGCOMM 1998, (Vancouver, Canada)*, September 1999.

[17] R. Pan, B. Prabhakar, K. Psounis, and D. Wischik. SHRiNK: A method for scaleable performance prediction and efficient network simulation. In *Proceedings of IEEE INFOCOM 2003, (San Francisco, California)*, March 2003.

[18] L. L. Perterson and B. S. Davie. *Computer Networks: A Systems Approach*. Morgan Kaufman, 1996.

[19] S. Shakkottai and R. Srikant. How Good are Deterministic Fluid Models of Internet Congestion Control? In *Proceedings of IEEE INFOCOM 2002, (New York, New York)*, June 2002.

[20] The Mathworks Inc. *MATLAB: The Language of Technical Computing*. The Mathworks Inc., 1999.

[21] Y. Wu and W. Gong. Time Stepped Simulation of Queuing Systems. In *Technical Report, Department of Electrical and Computer Engineering, University of Massachusetts*, 2001.