Network Calculus Based Simulation for TCP Congestion Control: Theorems, Implementation and Evaluation

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

Abstract—
In this paper, we examine the feasibility of incorporating network calculus based models in simulating TCP/IP networks. By exploiting network calculus properties, we characterize how TCP congestion control — additive increase and multiplicative decrease (AIMD), together with the queue management strategy used in routers, regulates TCP flows. We first divide the time axis into intervals (each of which consists of multiple round-trip times), and derive a TCP AIMD throughput model which derives the attainable throughput of a TCP flow, given the number of collisions in an interval. Then based on the derived throughput model, we define a set of network calculus based theorems that give upper and lower bounds on the attainable TCP throughput in each interval. Finally, we implement network calculus (NC) based simulation in *ns-2*, conduct simulation in both the packet mode and the network calculus-based mode, and measure the performance gain (in terms of the execution time thus reduced) and the error discrepancy (in terms of the discrepancy between NC-based simulation results and packet-level simulation results).

The simulation results indicate an order of magnitude or more (maximally 30 times) improvement in execution time and the performance improvement becomes more pronounced as the network size increases (in perspective of network capacities and number of flows). The error discrepancy between NC-based simulation and packet-level simulation, on the other hand, is minimally 1-2% and maximally 8-12% in a wide spectrum of network topologies and traffic loads employed in this study. The simulation results also indicate the superiority of NC-based simulation over fluid model based simulation (the latter realized using the time stepped hybrid simulation).

Keywords—**Network calculus, performance modeling and simulation.**

I. Introduction

Modern data communication networks are extremely complex and do not lend well to theoretical analysis. It is common that network analysis can be rigorously made after leaving out several (sometimes subtle) details that cannot be easily captured in the analysis [6], [16], [18], [19]. Instead packet-level, event-driven simulation studies are usually carried out to better study the performance of network components, algorithms and protocols, and their interaction. The main obstacle in packet-level network simulation is, however, the vast number of packets that have to be simulated in order to produce accurate results. Each packet will generate a number of events (e.g., arrival of a packet at the router, its departure, and the event that a queue becomes empty, 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, for simulating large-scale networks.

What seems to be reasonable is really to combine theoretical modeling with packet-level simulation so as to exploit the advantages of both techniques, while not significantly suffering from their drawbacks. The main intent of this work is thus to examine the feasibility of leveraging theoretical theories in large scale network simulation. Specifically we focus on employing *network calculus* to model TCP/IP-operated networks and incorporating end results into simulations.

The idea of leveraging theoretical results in large scale network simulation is not new. Several researchers have proposed the notion of fluid model based simulation to mitigate the computational overhead in packet level simulation [13], [15], [19], [20]. Conceptually, they adopted and incorporated into simulation a fluid model — a set of differential equations (such as those derived in [17]) that characterize the network dynamics governed under the protocols in study. In the course of simulation, they then "convert" a sequence of (usually) closely-spaced packets into a fluid chunk (characterized by the fluid rate) and utilize the fluid model to obtain the parameters of interest (e.g., system throughput), rather than carrying out packet-level simulation. In spite of its effectiveness in reducing the execution time incurred in simulation, fluid model based simulation may not be well-suited in the case of light and/or sporadic traffic. This is because fluid models are derived based on the assumption of existence of a large number of flows [14], [17], [19]. To tackle this problem, we propose to use network calculus models as an alternative to fluid models. Note that network calculus is grounded on the mathematical theory of *Min-Plus* (or *Max-Plus* algebra) [1], [5], [7] and does not usually make the assumption of existence of a large number of flows.

Specifically, we examine the feasibility of incorporating network calculus based models in simulating TCP/IP networks. By exploiting network calculus properties, we characterize how TCP congestion control — additive increase and multiplicative decrease (AIMD), together with the queue management strategy used in routers, regulates TCP flows. Following the same line of approaches as in fluid model based simulation [11], we first divide the time axis into intervals (each of which consists of multiple round-trip times), and derive a TCP AIMD throughput model which derives the attainable throughput of a TCP flow, given the number of collisions in an interval. Then based on the derived throughput model, we define a set of network calculus based theorems that give upper and lower bounds on the attainable TCP throughput in each interval. Finally, we implement network calculus (NC) based simulation in *ns-2*, conduct simulation in both the packet mode and the network calculus-based mode, and measure the performance gain (in terms of the execution time thus reduced) and the error discrepancy (in terms of the discrepancy between NC-based simulation results and packet-level simulation results). The simulation results indicate an order of magnitude or more (maximally 30 times) improvement in execution time and the performance improvement becomes more pronounced as the network size increases (in perspective of network capacities and number of flows). The error discrepancy between NC-based simulation and packet-level simulation, on the other hand, is minimally 1-2% and maximally 8-12%, compared to bottleneck link capacity, in a wide spectrum of network topologies and traffic loads employed in the experiments. The simulation results also indicate the superiority of NC-based simulation over fluid model based simulation with time stepped hybrid simulation technique [11]. We chose the latter as the baseline for comparison because fluid model based simulation with the time stepped hybrid technique actually sends, receives, and drops packets, and carries out other protocol activities instead of solving differential equations at given time points. In several network configurations, we found that fluid model based simulation not only incurs larger execution time, but also give an error discrepancy (in the range of 5%–70%), which is several orders of magnitude larger than that in NC-based simulation.

Based on the simulation results, we conclude that in addition to their existing applications in the traffic regulation area, network calculus based models are useful tools for improving the performance of network simulation. To the best of our knowledge, this is the first attempt of exploring the use of network calculus in large-scale network simulation.

The rest of the paper is organized as follows. In Section II, we give a summary of existing work on fluid model based simulation and give a succinct introduction on *network calculus* theory. In Section III, we develop two analytical models: one derives the minimally or maximally achievable throughput and the other defines rules for emulating TCP flows in simulation engines. Following that we elaborate on how to implement NC-based simulation, and present our simulation results in Section IV. Finally we conclude the paper in Section V.

II. Preliminaries

In this section we summarize existing work in fluid model based simulation and give an overview of network calculus theory, in particular, its theorems and notations that pertain to our work.

A. Existing Work in Fluid Model Based Simulation

Several research efforts have focused on fluid model based simulation. Liu *et al.* [13] 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.* [15] 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.* [20] 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 [17], [18], [19]. Liu *et.al.* [14] 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.

However, as indicated in Section I, fluid model based simulation may not be well-suited in the case of light and/or sporadic traffic, because of the underlying assumption existence of a large number of flows [14], [17], [19]. To further demonstrate this, we carry out a simulation study the network configurations given in Figs. 5 and 7. One set of simulation is carried out in a simple network (Figs. 5) with the number of sender nodes varying from 10 to 60. The link bandwidth and delay are labeled in the figure, and the buffer size at each router is 50 packets. As given in the first and third columns of Table IV, the error discrepancy between fluid model based simulation and packet-level simulation is significant in the case that the number of flows is small.

B. Network Calculus and its Notations

Cruz analyzed network delays on the basis that data streams that enter the network satisfy "burstiness constraints" [9], [10]. Such an analysis lays its base on *Min-Plus* algebra [1], [5], [7] and has been recognized as constrained traffic regulation problems. The fundamental components developed in the system theory are *shaper* (or *regulator*, or *buffered leaky bucket controller*) for traffic regulation and *service curve* (or *f-server*) for provisioning of service guarantee. The results derived in the theory enable one to understand fundamental properties of, and hence better deploy, flow control, multimedia smoothing, delay control, and integrated service provisioning. Additionally, by extending the time invariant theory to the time varying case, dynamic traffic regulation or service guarantee can be considered in the same framework [5], [8]. Finally, by introducing packetizers, the effect of variable length packets can be analyzed using the theory developed under network calculus [4], [7]. The interested reader is referred to [1], [5], [7] for a detailed account of network calculus.

Although network calculus has been mainly used to study fundamental properties of QoS control, it has been recently applied to other domains. In particular, Baccelli and Hong [2] show that key feature of TCP operated networks can be expressed via a linear dynamical system in the max-plus algebra (max-plus algebra is the dual form of min-plus algebra). Inspired by their insight, we would like to investigate whether or not, and how, NC-based models can be used to facilitate large-scale network simulation.

Before delving into the derivation, we first introduce several important operations and notations that pertain to our work. In the network calculus theory a data flow is usually described by means of the cumulative function $f(t)$ or $g(t)$, defined as the number of bits seen on the flow in the time interval $[0, t]$. These functions belong to F, the set of wide-sense increasing sequences or functions f taking values in $R^+ \cup \infty$, (which are zero for non positive values) [5], [7]. The following operations are defined as basic operations under min-plus algebra.

- $\wedge$ represents the minimum operation, $f \wedge g = \min[f, g]$;
- $\vee$ represents the maximum operation, $f \vee g = \max[f, g]$;
- $\otimes$ represents the min-plus convolution of two functions or sequences $f, g \in F$,
$(f \otimes g)(t) = \inf_{0 \le s \le t} [f(t-s) + g(s)]$;
- $\overline{f}$, for $f \in F$ represents the sub-additive closure of f, which is $\overline{f} = \inf_{n \ge 0} \left[f^{(n)}\right]$.

To extend the min-plus algebra to the time varying case, the family of bivariate functions is introduced: $\tilde{F} = \{F(\cdot,\cdot) : F(s,t) \ge 0, F(s,t) \le F(s,t+1), \text{ for all } 0 \le s \le t\}$, and if $F(\cdot,\cdot) \in \tilde{F}$, $F(t,t) = 0$ for all t. Also, the following functions are defined for $\tilde{F}$:

- $(F \wedge G)(s,t) = \min[F(s,t), G(s,t)]$;
- $(F \vee G)(s,t) = \max[F(s,t), G(s,t)]$;
- $(F \otimes G)(s,t) = \inf_{s \le \tau \le t}\{f(G,\tau) + G(\tau,t)\}$;
- $\overline{F}(s,t)$ represents the sub-additive closure of $F(s,t)$,
$\overline{F}(s,t) = \inf_S \sum_{i=1}^{m} [F(t_{i-1}, t_i)]$, where $S = t_0, t_1, t_2, \ldots, t_m$ is any subset of {1,2,...,t} with $t_0 = s < t_1 < t_2 < \ldots < t_m = t$.

III. Analytic Model

In this section, we derive two throughput models: the throughput model and the network calculus model. In the throughput model, we derive the throughput attained in a time

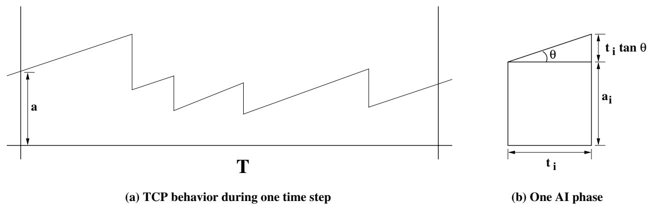


Fig. 1. A typical histogram of the attainable throughput under the TCP AIMD mechanism in an interval T, where a is the initial window size in the given period, and t_{R_i} is the round trip time.

interval, T, under the TCP AIMD congestion control mechanism. The interval T will be used as the value of the time step in simulation. Then, based on the derived throughput model, we develop the network calculus model that gives upper and lower bounds on the attainable TCP throughput in each interval. Finally, we derive the other functions (i.e., the queue, loss, and output functions needed to realize network calculus-based simulation.

A. Throughput Model

We denote, respectively, the additive increase parameter and the multiplicative decrease parameter under TCP AIMD as α and β. We make the following assumptions in the derivation: (i) all the routers in the network employ FIFO (drop tail) as their buffer occupancy discipline, and (ii) the round trip time, t_{R_i}, *within an interval* T is constant. (Note that we do not make the assumption that t_{R_i} is constant *throughout* the simulation.)

We intend to capture the throughput dynamics under the TCP AIMD mechanism in an interval T. Let the number of packet losses in the interval T be $m-1$, and the initial window size be a. Fig. 1 depicts a typical histogram of the attainable throughput under TCP AIMD in an interval T ((a)) and in an AI phase ((b)). Under the assumption that the round trip time is constant in each interval T, the slope in each AI phase in the interval T is equal to $\tan\theta_T = \alpha\frac{1}{t^2_{R_i}}$, by Kelly's deterministic differential equation, $\dot{r} = \alpha\frac{1}{t^2_{R_i}} - \beta r(t-t_{R_i})\cdot p(t-t_{R_i})$, that characterizes the TCP throughput dynamics, (where $p(t)$ is the packet loss probability function [12]).

Given $m-1$ packet losses in an interval T, we have m AI phases within that interval, and the total attainable throughput can be expressed as

$$\begin{aligned} S_{T,m-1} &= \sum_{i=1}^{m} s_i \\ &= a\cdot\left(t_1+\beta t_2+\beta^2 t_3+\ldots+\beta^{m-1}t_m\right) \\ &+ \tan\theta_T\cdot t_1\cdot\left(\frac{1}{2}t_1+\beta t_2+\ldots+\beta^{m-1}t_m\right) \\ &+ \cdots \\ &+ \tan\theta_T\cdot t_m\cdot\left(\frac{1}{2}t_m\right), \end{aligned} \quad (1)$$

where $t_i = a_i\cdot t_{R_i}$, $a_i \geq 1$ and $a_i \in Z$, for $1 \leq i \leq m$, and s_i stands for the throughput of each AI phase within T.

(1) can be recast in the vector and matrix notation as

$$S_{T,m-1} = \tan\theta_T\cdot t^T\cdot Q\cdot t + a\cdot P^T\cdot t,^1 \quad (2)$$

^{1}The superscript symbol T represents transposition.

where

$$\begin{aligned} t &= \begin{pmatrix} t_1 & t_2 & t_3 & \ldots & t_{m-1} & t_m \end{pmatrix}^T, \\ Q &= \begin{pmatrix} \frac{1}{2} & \beta & \beta^2 & \ldots & \beta^{m-2} & \beta^{m-1} \\ 0 & \frac{1}{2} & \beta & \beta^2 & \ldots & \beta^{m-2} \\ \cdots & & & \cdots & & \cdots \\ 0 & 0 & 0 & \ldots & \frac{1}{2} & \beta \\ 0 & 0 & 0 & \ldots & 0 & \frac{1}{2} \end{pmatrix}, \\ P &= \begin{pmatrix} 1 & \beta & \beta^2 & \ldots & \beta^{m-2} & \beta^{m-1} \end{pmatrix}^T. \end{aligned}$$

Minimum throughput. We derive the minimum and maximum attainable throughput for a tagged TCP flow when the interval, T, and, the number of collisions, $m-1$, are given. We formulate the problem of deriving the minimum attainable throughput as follows:

$$\begin{aligned} \min \quad & S_{T,m} = \tan\theta_T\cdot t^T\cdot Q\cdot t + a\cdot P^T\cdot t \\ \text{subject to} & \sum_{i=1}^{m} t_i = T. \end{aligned} \quad (3)$$

In order to solve the problem, we first investigate (with the use of the Lagrange multiplier theory [3]) whether or not a minimum solution exists, and if so, we can use one of the Lagrange multiplier algorithms [3] to obtain the solution. We formulate the Lagrangian function L as follows:

$$L(t,\lambda) = S_{T,m}(t) + \lambda h(t), \quad (4)$$

where $h(t) = \sum_{i=1}^{m} t_i - T = 0$.

The first and second derivatives of (4) are, respectively,

$$\begin{aligned} \bigtriangledown_t L(t,\lambda) &= \tan\theta_T\cdot Q'\cdot t + a\cdot P + \lambda, \quad (5) \\ \bigtriangledown_\lambda L(t,\lambda) &= \sum_{i=1}^{m} t_i - T, \quad (6) \\ \bigtriangledown^2_{tt} L(t,\lambda) &= \tan\theta_T\cdot Q', \quad (7) \end{aligned}$$

where

$$Q' = \begin{pmatrix} 1 & \beta & \beta^2 & \ldots & \beta^{m-2} & \beta^{m-1} \\ \beta & 1 & \beta & \beta^2 & \ldots & \beta^{m-2} \\ \beta^2 & \beta & 1 & \beta & \ldots & \beta^{m-3} \\ \cdots & \beta^2 & \beta & \cdots & \beta & \cdots \\ \beta^{m-2} & \beta^{m-3} & \ldots & \beta & 1 & \beta \\ \beta^{m-1} & \beta^{m-2} & \beta^{m-3} & \ldots & \beta & 1 \end{pmatrix}. \quad (8)$$

As the matrix Q' is positive definite since $\det(Q') > 0$ and $\tan\theta_T > 0$ since $0 < \theta_T < 90^o$, there exists a minimum solution of S_T, subject to $\sum_{i=1} t_i = T$.

Maximum throughput. Similarly we formulate the problem of deriving the maximum attainable throughput as

$$\begin{aligned} \min \quad & S'_{T,m} = -S_{T,m} \\ & = -(\tan\theta_T\cdot t^T\cdot Q\cdot t + a\cdot P^T\cdot t) \\ \text{subject to} & \sum_{i=1}^{m} t_i = T. \end{aligned} \quad (9)$$

Following the same line of derivation as in the minimization problem, we define the Lagrangian function L' for this maximization problem as follows:

$$L'(t,\lambda) = -(\tan\theta_T\cdot t^T\cdot Q\cdot t + a\cdot P^T\cdot t) + \lambda h(t), \quad (10)$$

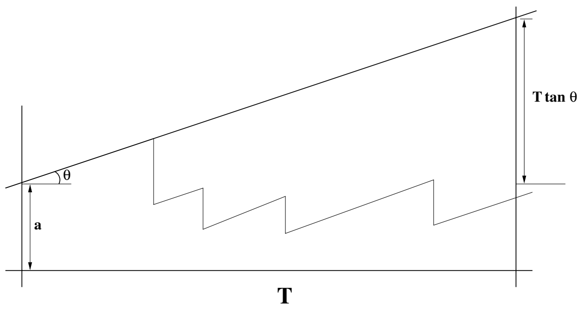


Fig. 2. The maximum attainable throughput in an interval of T.

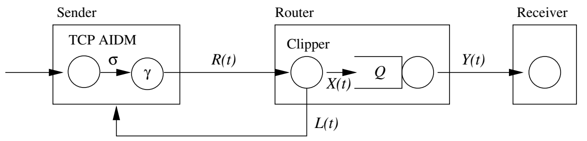


Fig. 3. The network configuration in the network calculus domain.

where $h(t) = \sum_{i=1}^{m} t_i - T = 0$.

The first and second derivatives of (10) are, respectively,

$$\begin{aligned} \bigtriangledown_t L'(t,\lambda) &= -(\tan\theta_T \cdot Q' \cdot t + a \cdot P) + \lambda, && (11)\\ \bigtriangledown_\lambda L'(t,\lambda) &= T - \sum_{i=1}^{m} t_i, && (12)\\ \bigtriangledown_{tt}^2 L'(t,\lambda) &= -\tan\theta_T \cdot Q', && (13) \end{aligned}$$

where Q' is given in (8).

Since the second derivative, $-\tan\theta_T \cdot Q$, of the Lagrangian function in (10) does not satisfy the second order condition for optimality in Lagrange multiplier theory, we cannot draw the conclusion of whether or not there exists a solution for this problem. However, by Wiestrass' Theorem [3], we know that the solution exists, and can be obtained when all the collisions occur at the beginning or ending point of the interval. Fig. 2 depicts such a scenario. That is, we can calculate the maximum attainable throughput in a given interval T as

$$S_T^{max} = \begin{cases} \left(a + \frac{1}{2}\tan\theta_T \cdot T\right) \cdot T, & \text{if } T > 0;\\ 0, & \text{if } T \le 0. \end{cases} \quad (14)$$

B. Network Calculus Model

Based on the above throughput model, we now derive the network calculus model that will be used to determine the TCP throughput in network simulation. Let R, X, Y, and L denote, respectively, the cumulative amount of the various types of traffic as labeled in Fig. 3. For example, $R(t)$ denotes the amount of traffic sent by the TCP connection in $[0, t]$. These functions belong to the set, F, of wide-sense increasing sequences or functions f taking values in $R^+ \cup \infty$ (Section II). Note that we take t in the discrete time domain in this study ($t \in Z$), and consider that the time axis is divided into intervals in the discrete time domain, $T_i = [t_i, t_{i+1}]$, $i \ge 0$, each of which in turn consists of multiple round-trip times of equal length. We also define

$$R_i(t) = R(t_i + t) - R(t_i), \text{ when } t \in T_i = [t_i, t_{i+1}]. \quad (15)$$

We first prove that TCP AIMD is a piecewise linear time varying shaper in the following theorems and corollary. The piecewise linear property in the shaper function means that the function is defined by concatenating several linear functions without overlapping, and the time varying property means that given a function of two time variables $H(t, s)$, the shaper forces the output $R(t)$ to satisfy the condition

$$R(t) \le H(s,t) + R(s)$$

for all $s \le t$.

Theorem 1: TCP AIMD is a piecewise linear time varying shaper with the time varying shaping curve σ_{T_i} in the interval $T_i = [t_i, t_{i+1}]$, where

$$\begin{aligned} \sigma_{T_i}(s,t) &= \sigma_{T_i}(t) - \sigma_{T_i}(s), && (16)\\ \sigma_{T_i}(t) &= \left(a_i + \frac{1}{2}\frac{\alpha}{t_{R_i}^2} t\right) t, &&\\ \text{and } a_i &= r(t_i) = R(t_i) - R(t_i - 1). && \end{aligned}$$

Proof: First we know that the function $\sigma_{T_i}(t)$ is convex and piecewise linear since t takes discrete time. Hence we can connect different successive points $\left(t, (a_i + \frac{1}{2}\frac{\alpha}{t_{R_i}^2} t)t\right)$ for all $t = 0, 1, 2, 3, \ldots$, and the resulting curve will be a concatenation of segments of slopes equal to $\frac{1}{2}\frac{\alpha}{t_{R_i}^2}(2t+1) + a_i$.

Let $R_i(t) = \sum_{i=1}^{t} r_i(t)$ be the cumulative throughput in the interval $[1, t]$, and $\hat{R}_i(s,t) = R_i(t) - R_i(s)$ be the cumulative throughput in the interval $[s+1, t]$, and $R_i(t) = 0$, for $t \le 0$. Then we have the following relation:

$$\hat{R}_i(s,t) = R_i(t) - R_i(s) = \sum_{k=s+1}^{t} r_k(t),$$
$$\text{for all } 0 \le s \le t \le (t_{i+1} - t_i).$$

From the previous analysis for maximum attainable throughput, we know that the throughput is upper-constrained by σ_{T_i} in the interval $T_i = [s, t]$ (Fig. 2 and (14)). Therefore, we have

$$\begin{aligned} R_i(t) - R_i(s) &= \sum_{k=s+1}^{t} r_k(t)\\ &\le \sigma_{T_i}(s,t)\\ &= \left(a + \frac{1}{2}\frac{\alpha}{t_{R_i}^2} t\right) t - \left(a + \frac{1}{2}\frac{\alpha}{t_{R_i}^2} s\right) s. \end{aligned}$$

Finally we have

$$\begin{aligned} R_i(t) &\le R_i(s) + \sigma_{T_i}(s,t),\\ R_i(t) &\le \min_{0\le s\le t} \left[R_i(s) + \sigma_{T_i}(s,t)\right]. && (17) \end{aligned}$$

By (17), we conclude that TCP AIMD is a piecewise linear time varying shaper with the time varying shaping curve σ_{T_i} in the interval T_i. ■

Theorem 2: TCP AIMD is upper constrained by a linear function $C(t)$ in any interval $T_i = [t_i, t_{i+1}]$, where C represents a given output capacity:

$$\begin{aligned} R_i(t) &\le R_i(s) + C(s,t),\\ R_i(t) &\le \min_{0\le s\le t} \left[R_i(s) + C \cdot (t - s)\right]. && (18) \end{aligned}$$

Proof: The proof for this theorem is similar to that for Theorem 1, and is then omitted. ■

Theorem 3: TCP AIMD is upper constrained by a linear function $W(t)$ in any interval $T_i = [t_i, t_{i+1}]$, where W represents a given maximum advertised window:

$$\begin{aligned} R_i(t) &\leq R_i(s) + W(s,t), \\ R_i(t) &\leq \min_{0\leq s\leq t} \left[R_i(s) + W\cdot(t-s)\right]. \end{aligned} \qquad (19)$$

Proof: The proof for this theorem is similar to that for Theorem 1, and is then omitted. ■

Corollary 1: The maximum solution of (17), (18), and (19), $R_i^{max}(t)$, is

$$\begin{aligned} R_i^{max}(t) &= (\overline{\sigma}_{T_i} \otimes (C \wedge W)(t), \qquad (20)\\ \text{where } \overline{\sigma}_{T_i}(t) &= a + \frac{1}{2}\frac{\alpha}{t_{R_i}^2}t. \end{aligned}$$

Proof: We note that the initial condition $R_i(t) = 0$, for $t \leq 0$. Also, following [5] we define the $\delta(t)$ function as

$$\delta_\tau = \begin{cases} \infty, & \text{if } t > \tau; \\ 0, & \text{if } t \leq \tau. \end{cases}$$

Therefore, $R_i(t)$ is constrained by

$$R_i(t) \leq (\sigma_{T_i} \otimes R_i)(t) \wedge C(t) \wedge W(t) \wedge \delta_0(t).$$

By Theorem 4.3.1 in [5] or Theorem 4.1.5 in [7],[2], we know that the maximal solution of the above equation is

$$R_i^{max}(t) = (\overline{\sigma}_{T_i} \otimes (C \wedge W))(t),$$
$$\text{where } \overline{\sigma}_{T_i}(t) = \min_{n\geq 0}\left[\sigma_{T_i}^{(n)}\right].$$

Note that the min-plus convolution, $\otimes$, of two convex functions is also convex. Furthermore, by the property stated/proved in the section entitled *properties of* $\otimes$ *for convex function* in [5], if the two functions are convex and piecewise linear, the min-plus convolution can be obtained by sorting the different linear pieces of the two functions in the order of increasing slopes. Now σ_{T_i} is convex and piecewise linear since it has a linear slope, $\frac{1}{2}\frac{\alpha}{t_{R_i}^2}(2t+1)+a$, in each interval $[t, t+1]$ for $t \geq 1$, except the initial slope in the interval $[0, 1]$, which is $a + \frac{1}{2}\frac{\alpha}{t_{R_i}^2}$. By using the aforementioned property, $\sigma_{T_i} \otimes \sigma_{T_i}$ can be obtained by doubling the length of different linear segments of σ_{T_i}, and sorting them end-to-end in the increasing order of their slopes. Note again that the first linear segment whose slope $a + \frac{\alpha}{t_{R_i}^2}$ has a double length. If we repeat this operation for convolution n times, we obtain a convex piecewise linear sequence $\sigma_{T_i}^{(n)}(t)$:

$$\sigma_{T_i}^{(n)}(t) = a + \frac{1}{2}\frac{\alpha}{t_{R_i}^2}t \qquad \text{if } \ 0 \leq t \leq n.$$

Conclusively, the sub-additive closure of σ_{T_i} is determined when $n \to \infty$, and therefore,

$$\overline{\sigma}_{T_i}(t) = a + \frac{1}{2}\frac{\alpha}{t_{R_i}^2}t.$$

■

In what follows we prove in the following Lemma and Theorem that TCP AIMD is a system that offers a time varying service curve γ_{T_i} in interval T_i. The piecewise linear property in the service curve means that the curve is defined by concatenating several linear functions without overlapping, and the time varying property means that given a function of two time variables $H(t,s)$, the curve serves the output $R(t)$ to satisfy the condition

$$R(t) \geq H(s,t) + R(s)$$

for all $s \leq t$.

Lemma 1: The function $\gamma_T(t_i, t_j)$ is non-negative and wide-sense increasing in $t \in T_i = [t_i, t_{i+1}]$, where

$$\begin{aligned} \gamma_{T_i}(t_i, t_j) &= \gamma_{T_i}(t_j) - \gamma_{T_i}(t_i), \qquad (21)\\ \text{where } \gamma_{T_i}(t) &= \min S_{t,m}, \text{ when } t \in T_i. \qquad (22) \end{aligned}$$

Proof: According to [5] and [7], if $\gamma_{T_i}(t_i, t_j) \geq 0$ and $\gamma_{T_i}(t_i, t_j) \leq \gamma_{T_i}(t_i, t_j + 1)$ for all $0 \leq t_i \leq t_j$, $\gamma_{T_i}(t_i, t_j)$ is a nonnegative and increasing in t. We know that $\gamma_{T_i}(t_i, t_j) \geq 0$ since $\min S_{(t_j - t_i),m}$ is always greater than or equal to 0 when $m \geq 0$, and $(t_j - t_i) > 0$. In addition, since $\gamma_{T_i}(t_i, t_j + 1) = \gamma_{T_i}(t_i, t_j) + \gamma_{T_i}(t_j, t_j + 1)$ and $\gamma_{T_i}(t_j, t_j + 1) = \min S_{1,\tilde{m}}$, for $\tilde{m} \geq 0$, is always non-negative, we know that $\gamma(t_i, t_j + 1) \geq \gamma(t_i, t_j)$. Hence $\gamma(t_i, t_j)$ is non-negative and increasing in t. ■

Theorem 4: When the number of packet losses, $m - 1$, in an interval $T_i = [t_i, t_{i+1}]$ is known, TCP AIMD is a system that offers a time varying service curve γ_{T_i} which is piecewise linear in the interval T_i, where $\gamma_{T_i}(t) = \min S_{t,m}$, when $t \in T_i$.

Proof: First let t^* be the solution of minarg $S_{T_i,m}$. Given the number of packet losses, we can separate T_i into m AI phases $T_{i,j} = [t_{i,j-1}, t_{i,j}]$ for $1 \leq j \leq m$, each of which has the throughput function

$$\begin{aligned} \gamma_{T_{i,j}}(t) &= \left(a_{i,j} + \frac{1}{2}\frac{\alpha}{t_{R_i}^2}t\right)t, \\ \text{where } a_{i,j} &= r(t_{i,j-1}) \text{ and } t_{i,j-1} \leq t \leq t_{i,j}. \end{aligned}$$

We know that each function $\gamma_{T_{i,j}}$ is convex and piecewise linear, and since the sum of convex functions is convex, $\gamma_{T_i}(t)$ is convex and piecewise linear. The remaining derivation is similar to that in Theorems 1 and 2, and finally we have:

$$\begin{aligned} R_i(t) &\geq R_i(s) + \gamma_{T_i}(s,t), \\ R_i(t) &\geq \min_{0\leq s\leq t}\left[R_i(s) + \gamma_{T_i}(s,t)\right]. \end{aligned} \qquad (23)$$

■

Corollary 2: The minimum solution of (23), $R_i^{min}(t)$, is

$$R_i^{min}(t) = \overline{\gamma}_{T_i}(t) = \sum_{j=1}^{m}\left\{a_{j-1} + \frac{1}{2}\frac{\alpha}{t_{R_i}^2}(t_{i,j} - t_{i,j-1})\right\}, \qquad (24)$$

where all the terms are defined in Theorem 4.

Proof: By (23), $R_i(t)$ is constrained by

$$R_i^{min}(t) = (\gamma_{T_i} \otimes R_i)(t).$$

From Theorem 4.3.1 in [5] or Theorem 4.1.5 in [7], the solution of the above equation is

$$\overline{\gamma}_{T_i}(t) = \min_{n\geq 0}\left[\gamma_{T_i}^{(n)}\right].$$

[2]The proof is given in [1]

Since $T_i = [t_i, t_{i+1}]$ consists of m sub-intervals, where $T_{i,j} = [t_{i,j-1}, t_{i,j}]$ for $1 \le j \le m$, has the throughput function

$$\gamma_{T_{i,j}}(t) = \left(a_{i,j-1} + \frac{1}{2}\frac{1}{t_{R_i}^2} t \right) t, \tag{25}$$

we know that $R_i(t)$ is served with the time varying service curve given by (25) in the sub-interval $T_{i,j}$. Let $x(\tau) \overset{\triangle}{=} R_i(t_{i,j-1} + \tau) - R_i(t_{i,j-1})$. Then,

$$x(\tau) \ge (\gamma_{T_{i,j}} \otimes x)(\tau) = \min_{0 \le \omega \le \tau} \left[\gamma_{T_{i,j}}(\tau, \omega) + x(\omega) \right].$$

By employing the same reasoning as in the proof of Corollary 1, the solution of the above equation is

$$x^{min}(\tau) = \overline{\gamma}_{T_{i,j}}(\tau), \text{ where } \overline{\gamma}_{T_{i,j}}(t) = a_{i,j-1} + \frac{1}{2}\frac{\alpha}{t_{R_i}^2} t.$$

By recasting the above two equations with the original notation, we obtain respectively

$$\begin{aligned} &R_i(t_{i,j-1} + \tau) - R_i(t_{i,j-1}) \\ &\quad \ge \min_{0 \le \omega \le \tau} \left[\gamma_{T_{i,j}}(\tau, \omega) + R_i(t_{i,j-1} + \omega) - R_i(t_{i,j-1}) \right], \\ &R_i^{min}(t_{i,j-1} + \tau) - R_i^{min}(t_{i,j-1}) = \overline{\gamma}_{T_{i,j}}(\tau). \end{aligned}$$

Then, we have

$$\begin{aligned} R_i(t) &\ge \min_{t_{i,j-1} \le s \le t} \left[\gamma_{T_{i,j}}(t, s) + R_i(s) \right], \\ R_i^{min}(t) &= \overline{\gamma}_{T_{i,j}}(t) + R_i^{min}(t_{i,j-1}), \\ &\quad \text{when } t_{i,j-1} \le t \le t_{i,j}. \end{aligned}$$

By repeating this procedure for each sub-interval, we obtain the minimum solution, $R_i^{min}(t)$:

$$\begin{aligned} R_i^{min}(t) = \overline{\gamma}_{T_i}(t) &= \sum_{j=1}^{m} \overline{\gamma}_{T_{i,j}}(t) \\ &= \sum_{j=1}^{m} \left\{ a_{j-1} + \frac{1}{2}\frac{\alpha}{t_{R_i}^2}(t_{i,j} - t_{i,j-1}) \right\}. \end{aligned}$$

■

Based on Theorems 1-4 and their subsequent Corollaries or Lemmas, we recast the TCP/IP operated network in the domain of network calculus (Fig. 3). Then, we can use the cumulative input function $R(t)$ to emulate the TCP AIMD throughput behavior with the time-stepping technique.

The following theorems extend the throughput derivation from within an interval to the entire simulation period.

Theorem 5: When t is in the interval $T_i = [t_i, t_{i+1}]$, $R(t)$ is constrained by the time varying shaping curve σ_{T_i}, and hence $R(t)$ is upper constrained as follows:

$$R(t) \le \min_{t_i \le s \le t} \left[\sigma_{T_i}(t, t_i) + R(s) \right], \text{ when } t \in T_i. \tag{26}$$

Proof: By (17), $R_i(t)$ is upper constrained by a time varying shaping curve $\sigma_{T_i}(t_i, t_{i+1})$ in the interval $T_i = [t_i, t_{i+1}]$. Recall (from (15)) that

$$R_i(\tau) = R(t_i + \tau) - R(t_i). \tag{27}$$

Since $R_i(\tau)$ is upper constrained by σ_{T_i},

$$R_i(\tau) \le (\sigma_{T_i} \otimes R_i)(\tau) = \min_{0 \le \omega \le \tau} \left[\sigma_{T_i}(\tau, \omega) + R_i(\omega) \right].$$

Therefore, by Corollary 1,

$$\begin{aligned} R_i^{max}(\tau) &= (\overline{\sigma}_{T_i} \otimes (C \wedge W))(\tau), \\ &\quad \text{where } \overline{\sigma}_{T_i}(t) = a_i + \frac{1}{2}\frac{\alpha}{t_{R_i}^2} t. \end{aligned}$$

Recasting the original notation into the above equation, we obtain

$$\begin{aligned} &R(t_i + \tau) - R(t_i) \le \\ &\qquad \min_{0 \le \omega \le \tau} \left[\sigma_{T_i}(\tau, \omega) + R(t_i + \omega) - R(t_i) \right], \\ &R^{max}(t_i + \tau) - R^{max}(t_i) = (\overline{\sigma}_{T_i} \otimes (C \wedge W))(\tau). \end{aligned}$$

Rearranging the terms, we have

$$\begin{aligned} R(t) &\le \min_{t_i \le s \le t} \left[\sigma_{T_i}(t, s) + R(s) \right], \\ R^{max}(t) &= (\overline{\sigma}_{T_i} \otimes (C \wedge W))(t) + R^{max}(t_i), \\ &\quad \text{when } t \in T_i = [t_i, t_{i+1}]. \end{aligned} \tag{28}$$

■

Theorem 6: When the cumulative input function $R(t)$ is in the interval $T_i = [t_i, t_{i+1}]$, it is served by a time varying service curve γ_{T_i}, and hence $R(t)$ is minimally guaranteed as follows:

$$R(t) \ge \min_{t_i \le s \le t} \left[\gamma_{T_i}(t, s) + R_i(s) \right], \text{ where } t \in T_i. \tag{29}$$

Proof: By (23), $R_i(t)$ is lower bounded by a time varying service curve $\gamma_{T_i}(t_i, t_{i+1})$ in the interval $T_i = [t_i, t_{i+1}]$. Using the expression of $R_i(\tau)$ given in (27), we have

$$R_i(\tau) \ge (\gamma_{T_i} \otimes R_i)(\tau) = \min_{0 \le \omega \le \tau} \left[\gamma_{T_i}(\tau, \omega) + R_i(\omega) \right].$$

Following the same line of reasoning as in Theorem 5, we obtain

$$\begin{aligned} R(t) &\ge \min_{t_i \le s \le t} \left[\gamma_{T_i}(t, s) + R(s) \right], \\ R^{min}(t) &= \overline{\gamma}_{T_i}(t) + R^{min}(t_i), \\ &\quad \text{when } t \in T_i = [t_i, t_{i+1}]. \end{aligned} \tag{30}$$

■

With Theorems 5–6, we determine the maximum value of $R(t)$ using Eq. (28) and the minimum value of $R(t)$ using Eq. (30) at each time step T. Note that any non-decreasing function $R(t)$ such that $R^{min}(t) \le R(t) \le R^{max}(t)$ can be a solution to emulate TCP AIMD throughput in the context of *network calculus*, and therefore we believe instead of employing Theorem 5–6, the algorithm of generating TCP traffic can be further improved. This is a subject of our ongoing investigation.

C. Queue, loss, and output function

We also express the output, queue length and loss function at each router during $T_i = [t_i, t_{i+1}]$, but briefly introduce these functions without specific details due to space limitation. Let N be the total number of flows routed to the same output link, l, at each router.

Then, we define cumulative output, queue length, and cumulative loss amount at the output link, l, are respectively determined as follows:

$$\begin{aligned} Y(t_{i+1}) &= Y(t_i) + \\ &\quad \min(\sum_{j=1}^{N} R^j(t_i, t_{i+1}), C_l(t_{i+1} - t_i)), \quad (31) \\ q(t_{i+1}) &= \min(Q, \\ &\quad [q(t_i) + R(t_i, t_{i+1}) - C_l(t_i, t_{i+1})]^+), \quad (32) \\ L(t_{i+1}) &= L(t_i) + \\ &\quad [q(t_i) + R(t_i, t_{i+1}) - C_l(t_i, t_{i+1}) - Q]^+ (33) \end{aligned}$$

where in (31) we use superscript to denote each flow among multiple flows, and C_l represents the capacity of output link.

Each TCP flow "occupies" the cumulative output, loss and queue length according to the proportion of its flow amount to the total amount of flows. Additionally, we have to consider the round trip time per flow since flows with shorter round trip times occupy a larger portion of output, loss, and queue functions, as compared to flows with longer round trip times. Based on this observation, a router divides flows into several classes according to their round trip time, and then distribute the amount of output, loss, and queue length to each class in the ascending order of round trip time until the amount of $(Y(t_{i+1}) - Y(t_i))$ is consumed.

Let $N'(\leq N)$ be the number of classes, and $Class(k), 1 \leq k \leq N'$ the kth class. The amount of traffic that the class can send during $T_i = [t_i, t_{i+1}]$ is:

$$\begin{aligned} &Y^{Class(k)}(t_{i+1}) = Y^{Class(k)}(t_i) \\ &\quad + (Y(t_{i+1}) - Y(t_i)) \cdot \frac{\sum_{k' \in Class(k)} R^{k'}(t_i, t_{i+1})}{\sum_{j=1}^{N} R^j(t_i, t_{i+1})} \end{aligned} \quad (34)$$

The queue length that the class occupies is also determined as:

$$q^{Class(k)}(t_{i+1}) = q(t_{i+1}) \cdot \frac{\sum_{k' \in Class(k)} R^{k'}(t_i, t_{i+1})}{\sum_{j=1}^{N} R^j(t_i, t_{i+1})} \quad (35)$$

Within each class, we determine the output, loss, and queue length function for each flow j as follows. First, the output function for flow j is

$$\begin{aligned} Y^j(t_{i+1}) &= Y^j(t_i) \\ &+ \left(Y^{Class(k)}(t_{i+1}) - Y^{Class(k)}(t_i)\right) \\ &\cdot \frac{R^j(t_i, t_{i+1})}{\sum_{k' \in Class(k)} R^{k'}(t_i, t_{i+1})}. \end{aligned} \quad (36)$$

Second, the queue length function for flow j is

$$\begin{aligned} &q^j(t_{i+1}) = q^j(t_i) \\ &\quad + q^{Class(k)}(t_{i+1}) \cdot \frac{R^j(t_i, t_{i+1})}{\sum_{k' \in Class(k)} R^{k'}(t_i, t_{i+1})}. \end{aligned} \quad (37)$$

Last, the loss function for flow j is

$$\begin{aligned} &L^j(t_{i+1}) = L^j(t_i) + \\ &[R^j(t_i, t_{i+1}) - (Y^j(t_{i+1}) - Y^j(t_i)) - q^j(t_{i+1})]^+ . \end{aligned} \quad (38)$$

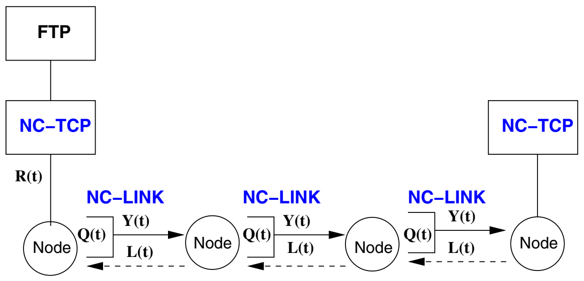


Fig. 4. Network components implemented for NC-based simulation.

IV. Simulation Study

In this section, we discuss how we incorporate network calculus results into network simulator *ns-2*. Based on the implementation, we also evaluate network calculus based (NC-based) simulation for TCP/IP-operated networks by conducting an extensive set of simulations.

A. Implementation

Based on all the equations derived in Section III, we implement *network calculus* based simulation in *ns-2*. In particular, we create two modules, *NC-TCP* and *NC-LINK*, and a new packet *nc-tcp*. Fig. 4 gives the software architecture that consists of *NC-TCP*, *NC-LINK*, and existing modules such as FTP module and nodes.

A *NC-TCP* module at the sending side performs the following tasks: it (i) directly supports Theorems 5 and 6, (ii) computes a round trip time used for the next time step on the basis of the loss and delay information fed back from the corresponding *NC-TCP* module, (iii) determines the maximum throughput and minimum throughput by (28) and (30), and then sends the computed amount of traffic via a *nc-tcp* packet at each time step. In the current implementation, we use the maximum throughput value in the case of no collision, and the minimum throughput value in the other case. We suspect that this approximation might aggravate the error discrepancy we will later present, and therefore we needs to improve this algorithm. A *NC-TCP* module at the receiving side simply computes the effective throughput when it receives *nc-tcp* packets, and responds with another *nc-tcp* packet with the loss and delay information collected over the forward path. This acknowledgment is delivered to the sending *NC-TCP* without intervention with *NC-LINK* since the backward path consists of existing *SimpleLink module* in *ns-2*.

NC-LINK modules are used to connect network nodes. They are equipped with drop-tail queue, can process *nc-tcp* packets and determine the output, loss, and queue function using (36), (38), and (37), respectively. *NC-TCP* operates in the time-stepped fashion [11] while *NC-LINK* is activated when all the flow information it handles arrives.

B. Simulation

We have conduct simulation to evaluate NC-based simulation and compare its performance (in terms of reduction in execution time and discrepancy between network calculus model based simulation and packet level simulation) against fluid-model based simulation, in a wide variety of network topologies. The packet size is set to 1000 bytes, unless otherwise stated. Two sets of experiments were carried out: one employs *TCP Tahoe* and the other uses *TCP Reno* for both packet level simulation and fluid model based simulations. Due to the space limit,

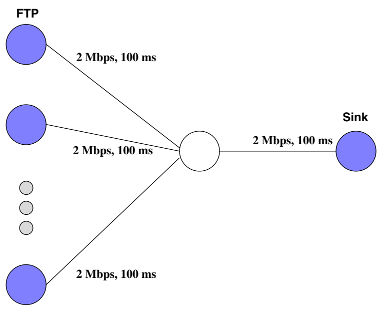


Fig. 5. The network configuration used in the first set of experiments.

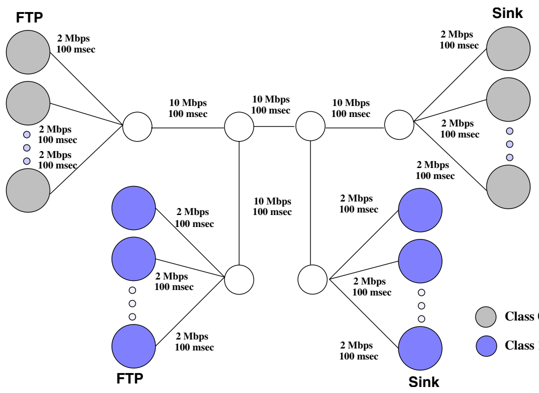


Fig. 7. The network configuration used in the second set of experiments.

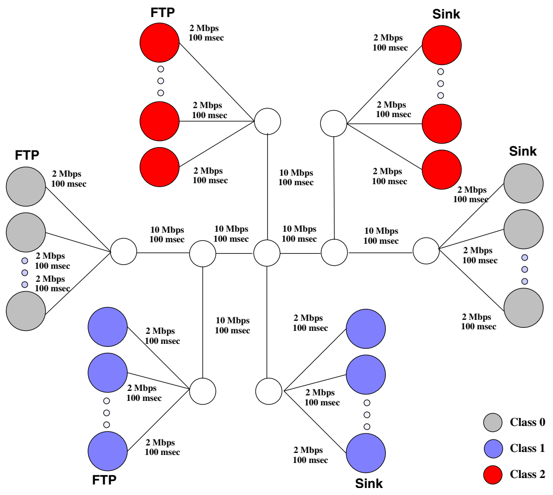


Fig. 9. The network configuration used in the third set of experiments.

in what follows we only present results with *TCP Reno*.

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, and *ns-2.1b9a* is chosen as the underlying simulation environment.

Performance of NC-based Simulation w.r.t. Error Discrepancy. First we examine how close the TCP throughput obtained under NC-based simulation is to that under packet level simulation, compared to bottleneck link capacity. The simulation is carried out in a simple network (Fig. 5) with the number of FTP nodes varying from 10 to 60. The link bandwidth and delay are labeled in the figure, and the buffer size at each router is 50 packets. Fig. 6 gives the number of packets delivered every 100 seconds in the system in an ℓ-second simulation under both simulation modes, when the number of nodes is 40 and the link bandwidth of each link is 2 Mbps ((a)) or 10 Mbps ((b)). Note that ℓ varies from 100 to 1000. As shown in Fig. 6, the results generated under NC-based simulation are very close to those under packet level simulation. The discrepancy in the results between the two simulation modes is maximally $\sim$10%.

To verify whether or not NC-based simulation still renders high-fidelity simulation results in more complicated network configurations, we repeat the same experiments but in more complicated network configurations. We first carry out the same experiments in the configuration given in Fig. 7, with the number of nodes in each class varying from 10 to 60, and the buffer size at each router being set to 200 packets. Fig. 8 depicts the number of packets delivered every 100 seconds in each class in an ℓ-second simulation under both simulation modes, when each of the two TCP classes consists of 20 nodes ((a)) and 40 nodes ((b)), respectively. Again NC-based simulation renders very close results to packet level simulation. As the number of flows increases, the discrepancy in the results between the two simulation modes also increases and is maximally $\sim$10%.

Fig. 9 gives another complicated network configuration where the buffer size at each router is 200 packets and the number of nodes in each class varies from 10 to 60. Fig. 10 gives the corresponding results. As shown in Fig. 10, a similar trend can be observed, except that the error discrepancy is now maximally $\sim$12%.

We believe the error discrepancy observed in the experimental results from the facts that (i) the simple algorithm to generate TCP packets in each *NC-TCP* connection uses the maximum throughput in the cases of no collisions and the minimum throughput in the other cases, and (ii) *network calculus* cannot fully approximate diverse TCP dynamics with linearity. However, if we compare two sets of results on a per flow basis, the observed error discrepancy is more acceptable.

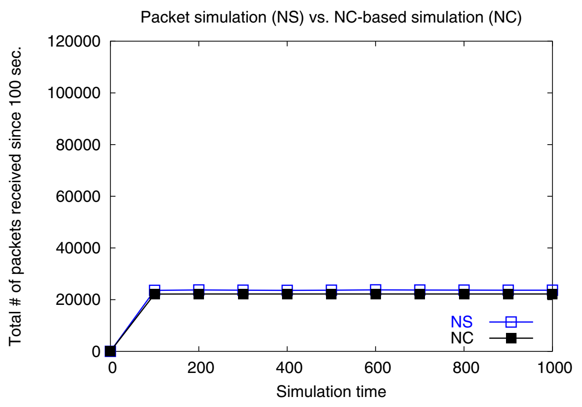
Performance of NC-based Simulation w.r.t. Execution Time. Fig. 11 gives the execution time it takes to carry out 1000-second simulation versus the number of nodes under packet level and NC-based simulation in the network configuration given in Fig. 5, while Fig. 12 (Fig. 13) gives the execution time taken to carry out 1000-second simulation versus the number of nodes per class in the network configuration given in Fig. 7 (Fig. 9).

As shown in Figs. 11–13, the execution time incurred in NC-based simulation is maximally 5-10 times less than that in packet level simulation. (The speed-up can be further improved with a larger time step value.)

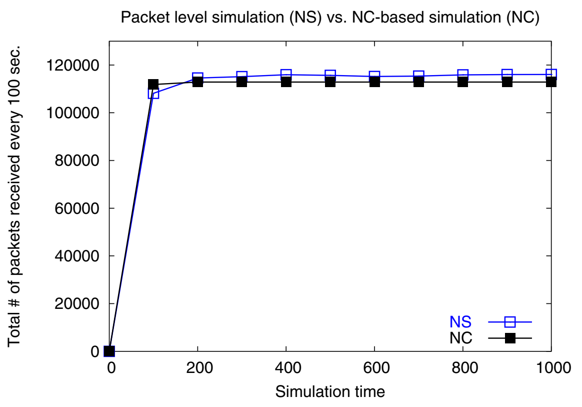
The reason why the execution time incurred in NC-based simulation slightly increases with the number of nodes per class results from the time step overhead. Even if the number of packets to be sent is small, the number of timing events in NC-based simulation increases with the number of nodes in the network configuration.

Effects of the Time Step Value. To investigate how the time step value, T, in Fig. 1, affects the performance in NC-based simulation, we carry out simulation that varies the time step value T. Tables I–II give the results for the network configurations given in Figs. 5 and 7, respectively. In both configurations, the buffer size is set to 50 packets.

As shown in the tables, the performance of NC-based simulation can be improved (up to about 20 times better than that of

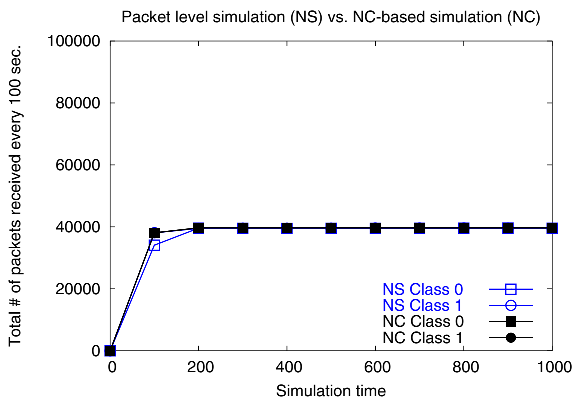


(a) Link bandwidth = 2 Mbps in Fig. 5

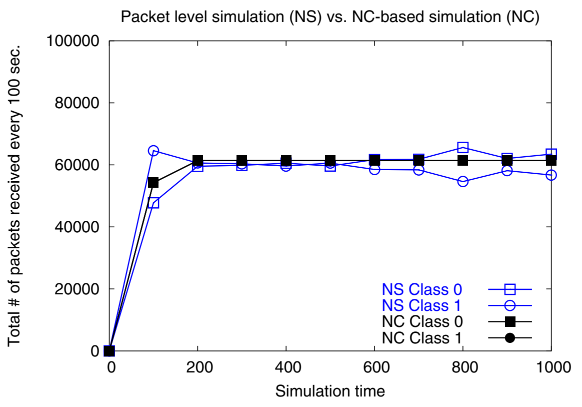


(b) Link bandwidth = 10 Mbps in Fig. 5

Fig. 6. The total number of packets received every 100 seconds under NC-based and and packet level simulation in an ℓ-second simulation, where ℓ varies from 100 to 1000. The network configuration is given in Fig. 5, where the number of nodes is 40.

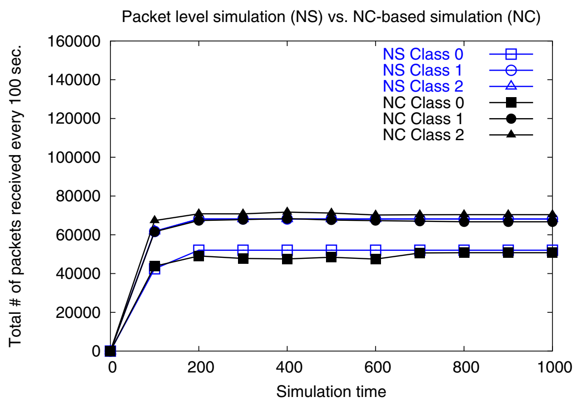


(a) # of nodes in each class = 20 nodes in Fig. 7

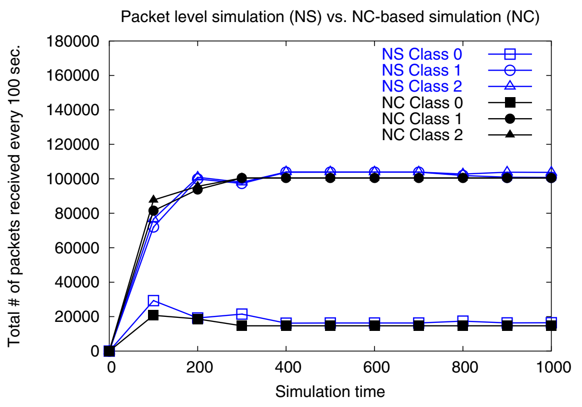


(b) # of nodes in each class = 40 nodes in Fig. 7

Fig. 8. The total number of packets received every 100 seconds under NC-based and packet level simulation in an ℓ-second simulation, where ℓ varies from 100 to 1000. The network configuration is given in Fig. 7.



(a) # of nodes in each class = 40 nodes in Fig. 7



(b) # of nodes in each class = 60 nodes in Fig. 7

Fig. 10. The total number of packets received every 100 seconds under NC-based and packet level simulation in an ℓ-second simulation, where ℓ varies from 100 to 1000. The network configuration is in Fig. 9.

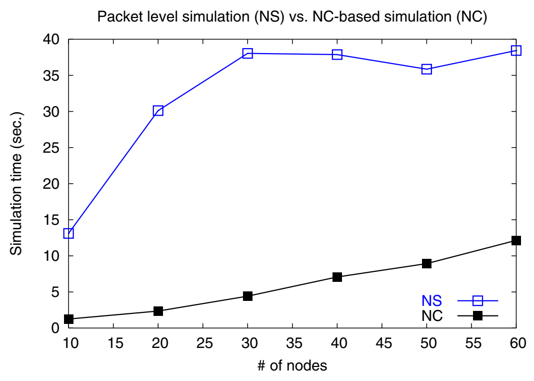


Fig. 11. The execution time taken to carry out 1000-second simulation versus the number of nodes, under packet level and NC-based simulation in the network configuration in Fig. 5, when the link bandwidth is *10 Mbps.*

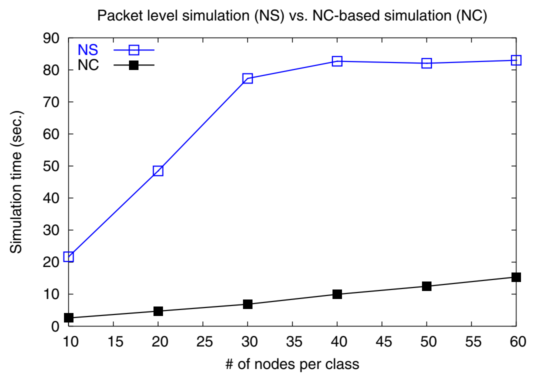


Fig. 12. The execution time taken to carry out 1000-second simulation versus the number of nodes per class, under packet level and NC-based simulation in the network configuration in Fig. 7.

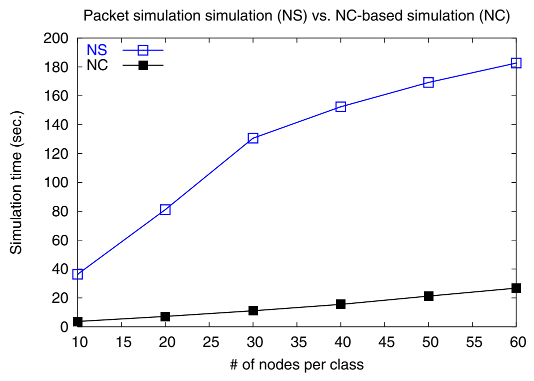


Fig. 13. The execution time taken to carry out 1000-second simulation versus the number of nodes per class, under packet level and NC-based simulation in the network configuration in Fig. 9.

TABLE I
THE NUMBER OF PACKETS RECEIVED IN A 1000-SECOND SIMULATION RUN AND EXECUTION TIME (SEC.) UNDER PACKET LEVEL AND NC-BASED SIMULATION IN THE NETWORK CONFIGURATION IN FIG. 5, WHERE THE LINK BANDWIDTH, DELAY, AND THE NUMBER OF NODES ARE *10 Mbps*, *100 msec.*, AND 40, RESPECTIVELY.

Time step value (sec.)	packet level simulation		NC-based simulation	
	total # of packets	time	total # of packets	time
	1148061	37.69		
0.5			1127576	9.78
1.0			1029433	6.58
2.0			1040768	5.38
4.0			918052	4.87

TABLE II
THE NUMBER OF PACKETS RECEIVED IN A 1000-SECOND SIMULATION RUN AND EXECUTION TIME (SEC.) UNDER PACKET LEVEL AND NC-BASED SIMULATION IN THE NETWORK CONFIGURATION IN FIG. 7, WHERE THE NUMBER OF NODES IS 20.

Time step value (sec.)	packet level simulation			NC-based simulation		
	class 1	class 2	time	class 1	class 2	time
	388598	390500	47.48			
0.5				396225	396225	13.81
1.0				395830	395830	7.63
2.0				395213	395213	4.57
4.0				394243	394243	3.04
8.0				392252	392252	2.33

packet level simulation) as the time step value increases. However, the error discrepancy also (slightly) increases. Also, the execution time cannot be unlimitedly reduced by increasing the time step value. This is because of the fact that in the current NC-based simulation implementation, we use matrix computation to determine, with the collision information, the number of packets to be sent (Section III). As the number of collisions linearly increases with the increase in the time step size, more computing time is needed when a large time step value is used.

Comparison with Fluid Model Based Simulation. We also compare NC-based simulation against fluid model based simulation. We use the time-stepped hybrid simulation (TSHS) technique [11] to realize fluid model based simulation, since instead of solving a set of differential equations for TCP dynamics, this technique actually sends, receives, drops, and acknowledges packets as other network simulators usually do. In TSHS, packets that arrive from the same session in each time step are grouped into a *chunk*, and all the packets in a chuck are assumed to be evenly spaced within the time step (the process of which is termed as *packet smoothing*). We use (31), (33) and (32), to compute the output, loss, and queue amount at each link module (output of router), and when packet losses occur, we distribute them evenly over all flows. We believe this technique has three major advantages in our implementation: (i) we do not randomly drop packets in a sequence of packets delivered in a *chunk* packet, but instead drop packets from the end of the packet sequence at each routing node. This improves TCP throughput, as it prevents TCP from retransmitting successfully received packets due to packets dropped with earlier sequence-number; (ii) we apply the time stepping technique only to TCP modules, so that no waiting time is incurred at nodes on the path. This prevents the round trip time from being prolonged as a result of using the time stepping technique in intermediate nodes; and (iii) we do not apply transmission, queuing, and propagation delay to data packets on the forward path, but to ACK packets. This improves accuracy in terms of throughput for fluid model based simulation.

Tables III–IV gives the results under packet level, NC-based, and fluid model based simulation in the network configuration given in Fig. 5 where the buffer size is 50 packets. The link bandwidth and delay used are, respectively, $\{$*2 Mbps*, *100 msec.*$\}$, and $\{$*10 Mbps 100msec.*$\}$. Note that we have attempted to tune the time step value to obtain the best results (error discrepancy and execution time) under both NC-based and fluid model based simulation. As compared to fluid model based simulation, NC-based simulation not only incurs (slightly) smaller execution time, but also more closely follows the trajectory under packet level simulation. Fluid model based simulation produces good results only when there exist a sufficiently large number of flows to saturate bottle-neck links.

Table V–VI gives the results under packet level, NC-based, and fluid model based simulation in the network configuration in Fig. 7 where the buffer size is 50 packets. (Note that this buffer size produces different results in Table V from ones in Fig. 8 and 12.) Table VI gives the results when all the link bandwidth are set to *10 Mbps* while Table V presents results obtained in the original configuration. Similar observations are made — NC-based simulation more closely follows the trajectory under packet level simulation.

In addition, we observe that the performance of fluid model based simulation is sensitive to the time step values. If the time step value is small, fluid model based simulation suffers from the excessive timeout events and cannot reduce the execution time as desired. On the other hand, if the time step value is large (especially when $n \times$ time step $\sim$ RTT), the error discrepancy becomes prohibitively larger. When TCP connections of different round trip times co-exist, it becomes a non-trivial task to tune the time-step value. Furthermore, NC-based simulation reduce the execution time more aggressively (30 times less than packet level simulation) when we choose larger time step values in these experiments.

Current Limitation and Further Improvement. As the number of flows (or the number of nodes) in all network configurations in the paper increases, NC-based simulation comes to have larger error discrepancy. The reason is as follows: as the number of flows increases, some flows, especially flows with longer round trip time, starves to use buffer facilities in the network, and henceforth, the flows cannot send any packet during the time step, T, but the current buffer model (see Section III-C) can serve all flows even at the extremely congested cases. Based on this observation, the exact solution is under development.

Additionally, by modifying the software architecture for implementing NC-based simulation from Fig. 4 to Fig. 14, we can further reduce the execution time by 10-30 %. (Due to page limitation, we do not show simulation results here.) This is because instead of feeding back *nc-tcp* packet (acknowledgment) at the receiver side, the *Loss Manager* module collects the loss information from routers and provides the information to the sender. As a result, the execution time is saved because of the reduction in the number of packet events to be processed.

V. Conclusion

In this paper, we study how effective *network calculus* (NC) based simulation is for simulating TCP/IP-operated networks. We first derive two analytical models to characterize TCP AIMD behavior in perspective of traffic loads: the throughput model that determines both the minimum and maximum throughput values of a TCP flow when the number of collisions in an interval is given, and the other is the network calculus model which, give based on the derived throughput model, the upper and lower bounds on the attainable TCP throughput in each interval. Finally, we implement network calculus (NC) based simulation in *ns-2*, conduct simulation in the packet mode, fluid model mode, and network calculus-based mode, and measure the performance gain and the error discrepancy.

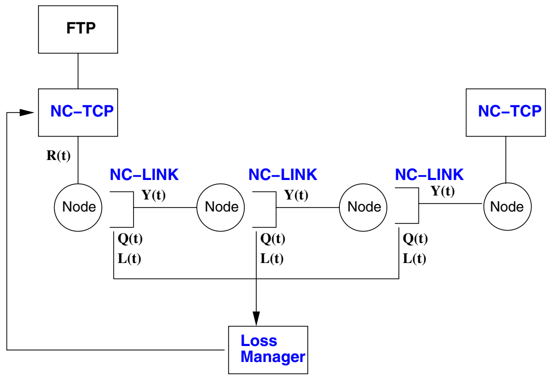


Fig. 14. A modified architecture for implementing NC-based simulation.

We show that NC-based simulation can give an order of magnitude or more (maximally 30 times) improvement in execution time, while yielding minimally 1-2% and maximally 8-12% error discrepancy, compared to bottleneck link capacity. Moreover, it outperforms significantly fluid model based simulation (realized using the time step hybrid simulation technique). Based on the simulation results, we conclude that in addition to their existing applications in the traffic regulation area, network calculus based models are useful tools for improving the performance of network simulation.

We have identified several directions for performance improvement: (i) as mentioned in Section IV, in the current implementation, we use, respectively, the maximum and minimum throughput values ((28) and (30)) to approximate the TCP throughput attained in an interval in the cases of no collision and collisions in that interval. This is perhaps over-simplified and a more elaborate version is under development; (ii) we plan to derive the packet dropping rules (e.g., those in Section III-C) at each link for other active queue management schemes in addition to make the current dropping rules more accurate; (iii) in the current implementation, the table operations (e.g., insert, delete, and search flow information) are implemented using sequential search. As a result, the search time increases dramatically as the number of flows increases. We expect the execution time incurred in NC-based simulation can be further reduced by employing priority-queue based search methods; (iv) we will seek better methods to solve the presented optimization problem; and (v) we will conduct more extensive experiments with extremely diverse network configurations to accurately analyze TCP dynamics within NC-based simulation.

At a higher level, we would like to apply NC-based simulation to other network architectures (in addition to TCP-operated networks). We will also look into the notion of mixed-mode simulation in which connection/protocol/network behaviors of interest are simulated at the packet mode while the rest of the system behavior is abstracted and simulated at the network calculus mode.

TABLE III

THE NUMBER OF PACKETS RECEIVED IN A 1000-SECOND SIMULATION RUN AND EXECUTION TIME (SEC.) UNDER PACKET LEVEL, NC-BASED, AND FLUID MODEL BASED SIMULATION IN THE NETWORK CONFIGURATION IN FIG. 5, WHERE THE LINK BANDWIDTH IS *2 Mbps* AND THE DELAY IS *100 msec.* (TIME STEP VALUE IN NC-BASED SIMULATION IS *1 sec.* WHILE THAT IN FLUID MODEL BASED SIMULATION IS *100 msec.*

# of nodes	packet level simulation		NC-based simulation		fluid simulation	
	total # of packets	time	total # of packets	time	total # of packets	time
10	239704	6.08	223296	1.42	138907	3.00
20	238499	6.78	244921	2.71	190110	4.96
40	236746	7.66	249218	5.00	241965	7.96

TABLE IV

THE NUMBER OF PACKETS RECEIVED IN A 1000-SECOND SIMULATION RUN AND EXECUTION TIME (SEC.) UNDER PACKET LEVEL, NC-BASED, AND FLUID MODEL BASED SIMULATION IN THE NETWORK CONFIGURATION IN FIG. 5, WHERE THE LINK BANDWIDTH IS *10 Mbps* AND THE DELAY IS *100 msec.* (TIME STEP IN NC-BASED SIMULATION IS *2 sec.* WHILE THAT IN FLUID MODEL BASED SIMULATION IS *100 msec.*)

# of nodes	packet level simulation		NC-based simulation		fluid simulation	
	total # of packets	time	total # of packets	time	total # of packets	time
10	495286	13.13	494462	0.78	154316	3.47
20	988174	30.61	988924	1.33	190811	4.96
40	1148061	37.20	1040768	5.38	408948	12.05

TABLE V

THE NUMBER OF PACKETS RECEIVED PER CLASS IN A 1000-SECOND SIMULATION RUN AND EXECUTION TIME (SEC.) UNDER PACKET LEVEL, NC-BASED, AND FLUID MODEL BASED SIMULATION IN THE NETWORK CONFIGURATION IN FIG. 7. (TIME STEP IN NC-BASED SIMULATION IS *2 sec.* WHILE THAT IN FLUID MODEL BASED SIMULATION IS *100 msec.*)

# of nodes per class	packet level simulation			network calculus simulation			fluid simulation		
	class 1	class 2	time	class 1	class 2	time	class 1	class 2	time
10	194336	196889	21.43	197606	197606	2.54	137528	114057	10.14
20	388598	390500	47.55	395213	395213	4.57	215649	309420	22.93
40	563605	569206	77.34	525735	525735	9.71	425703	537156	48.32

TABLE VI

THE NUMBER OF PACKETS RECEIVED PER CLASS IN A 1000-SECOND SIMULATION RUN AND EXECUTION TIME (SEC.) UNDER PACKET LEVEL, NC-BASED, AND FLUID MODEL BASED SIMULATION IN THE NETWORK CONFIGURATION IN FIG. 7 EXCEPT THAT ALL THE LINK BANDWIDTH ARE SET TO *10 Mbps*. (TIME STEP IN NC-BASED SIMULATION IS *2 sec.* WHILE THAT IN FLUID MODEL BASED SIMULATION IS *100 msec.*)

# of nodes per class	packet level simulation			network calculus simulation			fluid simulation		
	class 1	class 2	time	class 1	class 2	time	class 1	class 2	time
10	196081	198492	21.53	198871	198871	2.54	128712	156772	11.44
20	391540	393428	47.71	397742	397742	4.57	215649	309420	23.71
40	534113	596307	75.61	565787	565787	9.81	425703	537156	47.74

REFERENCES

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

[2] F. Baccelli and D. Hong. Tcp is max-plus linear: and what it telss us on its throughput. In *Proceedings of ACM SIGCOMM (Stockholm, Sweden)*, August 2000.

[3] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

[4] J.-Y. L. Boudec. Some properties of variable length packet shaper. In *Proceedings of ACM SIGMETRIC (Cambridge, Massachusetts)*, June 2001.

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

[6] F. Calí, M. Conti, and E. Gregori. Tuning of the ieee 802.11 protocol to achieve a theoretical throughput limit. *IEEE/ACM Transactions on Networking*, 8(6), December 2000.

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

[8] C.-S. Chang, R. L. Cruz, J. Y. Boudec, and P. Thiran. A min-plus system theory for constrained traffic regulation and dynamic service guarantees. In *Technical Report SSc/1994/024 EPFL*, July 1999.

[9] R. L. Cruz. A calculus for network delay, part i: Network elements in isolation. *IEEE Transactions on Information Theory*, 37(1), 1991.

[10] R. L. Cruz. A calculus for network delay, part ii: Network analysis. *IEEE Transactions on Information Theory*, 37(1), 1991.

[11] Y. Guo, W. Gong, and D. Towsley. Time-stepped hybrid simulation (tshs) for large scale networks. In *Proceedings of IEEE INFOCOM 2000 (Tel-Aviv Israel)*, March 2000.

[12] F. P. Kelly, A. Maulloo, and D. Tan. Rate control in communication networks: Shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49, 1998.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[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] Y. Wu and W. Gong. Time stepped simulation of queuing systems. In *Technical Report, Department of Electrical and Computer Engineering, University of Massachusetts*, 2001.