# Analysis of a TCP hybrid model*
## (Extended Version)
## Technical Report

Stephan Bohacek[†] João P. Hespanha[‡] Junsoo Lee[††] Katia Obraczka[§]

bohacek@math.usc.edu hespanha@usc.edu junsoole@usc.edu katia@cse.ucsc.edu

[†]*Department of Mathematics, Univ. of Southern California*
Los Angeles, CA 90089-1113

[‡]*Dept. Electrical Engineering–Systems, Univ. of Southern California*
Los Angeles, CA 90089-2563

[††]*Computer Science Department, Univ. of Southern California*
Los Angeles, CA 90089

[§]*Computer Engineering Department, University of California*
Santa Cruz, CA 95064

**Abstract**

In this paper we use hybrid systems to model and analyze the transient and steady-state behavior of multiple TCP flows that share a single common bottleneck link. The main contributions of this paper are (1) a characterization of the transient behavior of the flows, predicting their experimentally observed exponential synchronization and (2) a characterization of the steady-state regimen that is significantly more accurate than existing ones, especially under heavy traffic. The transient analysis of TCP also provides rules for network provisioning.

# 1 Introduction

For the past decade, TCP congestion control mechanisms have been under the scrutiny of the network research community. The existence of several versions of TCP such as TCP-Tahoe, Reno, Vegas, New Reno, and Selective Acknowledgement (SACK) is evidence of the attention TCP has received over the years. More recently, motivated by the increasing popularity of multimedia services, several efforts have been investigating *TCP-friendly* approaches to congestion control [1, 2, 3, 4]. One goal of TCP-friendly congestion control is to avoid the large window size variations that may be experienced by TCP flows and, at the same time, be able to coexist with TCP in a mutually fair way. This prompted several researchers to propose models that would permit the analysis of TCP and establish conditions for fairness with respect to alternative forms of congestion control.

Similarly to existing work on TCP congestion control, the model presented here is based on the *dumbbell topology.* In this topology (cf. Figure 1), $n$ persistent TCP flows

*This research was supported by NSF and DARPA

are generated at a source node $n_1$ and are directed towards a sink node $n_2$. All the flows compete for the finite bandwidth $B$ that characterizes the *bottleneck link* $\ell$ connecting the nodes. In more realistic networks, a path of several links (and intermediate nodes) would connect the source and destination. However, to analyze congestion control mechanisms, one often ignores the existence of all the intermediate links, except for the bottleneck link, i.e., the most congested link.

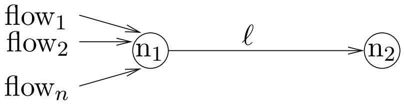


Figure 1: Dumbbell topology

This topology has been well studied in the context of TCP congestion control [5, 6, 7, 8, 9, 2, 3, 10] mainly for the two following reasons. First, for congestion control analysis purposes, it is believed to be (to some extent) representative of the behavior of a more general topology. In fact, while TCP behaves differently in different topologies, some characteristics are common to many of them and, hence, a complete understanding of how TCP behaves in one topology provides insight into how it might behave in other topologies. Second, analysis of more complicated topologies quickly becomes intractable. This problem of intractability has been accommodated by relying purely on simulation or, for theoretical work, on strong and often unrealistic assumptions. In this investigation, the framework of hybrid systems is utilized to theoretically determine specific properties of TCP without the use of overly simplifying assumptions. These properties are then validated through packet-level simulations performed using the network simulator `ns-2` [11].

Many of the recently proposed TCP-friendly algorithms are based on the well known relationship

$$T = \frac{1.23}{\overline{RTT}\sqrt{p}} \tag{1}$$

where $T$ is the average throughput, $\overline{RTT}$ the average round-trip time, and $p$ the average drop rate [7, 12, 13, 8, 14] or variations of (1) that consider timeouts [15, 3]. Our hybrid model provides a relationship between these quantities that, under heavy traffic, becomes significantly distinct from (1). The new relationship appears to be more accurate when compared with detailed packet-level simulations. One key difference is that our analysis does not require some of the simplifying assumptions found in previous work. For example, we do not assume that the round-trip time is constant. As it is well known, the round-trip time plays an important role in TCP: when the queue fills, the round-trip time increases and the TCP congestion window increases more slowly. In essence, the round-trip time has a stabilizing effect on the TCP flows, even *before* a drop has occurred.

The transient analysis of TCP also provides rules for network provisioning. For example, we show that the queueing capacity at the bottleneck should not be smaller than the "wire-capacity." In particular, that the maximum queue size $q_{\max}$ at the bottleneck link should be larger than $BT^\ell$, where $B$ denotes the link bandwidth and $T^\ell$ the round-trip propagation time. When this condition is violated, the bottleneck queue will empty, leading to underutilization of the available bandwidth.

### The philosophy behind a hybrid systems modeling framework

A few comments should be made regarding the philosophy behind the type of network modeling that we propose here. The most accurate network models are packet-level models that keep track of individual packets as they travel across the network. These are used in network simulators such as `ns-2` [11]. These models have two drawbacks: the large computational requirements for large-scale simulations and the difficulty in understanding how network parameters affect the overall system performance. Aggregate fluid-like models overcome these difficulties by simply keeping track of the average quantities that are relevant for network design and provisioning (such as queue sizes, transmission rates, drop rates, etc.) (cf. ,e.g., [14, 16]). The main limitation of these models is that they mostly capture steady state behavior and ignore the detailed transient behavior of congestion control because the averaging is typically done over large time scales. These models are unsuitable, e.g., to capture the dynamics of short-lived flow.

The model proposed in this paper fills the gap between packet-level and aggregate models by averaging discrete variables over a very short time scale (on the order of a round-trip time). This means that the model will be able to capture the dynamics of transient phenomena fairly accurately, as long as their time constants are larger than a couple of round-trip times. This is quite appropriate, e.g., for the analysis and design of congestion control mechanisms.

The "hybridness" of the model comes from the fact that, because of averaging, many variables that are essentially discrete (such as queue and window sizes) are allowed to take continuous values. However, because the averaging occurs over short time intervals, one still models discrete events such as the occurrence of a drop and the consequent reaction by congestion control. Also, one can still model fairly accurately the several distinct modes of TCP congestion control (slow-start, congestion avoidance, fast recovery, etc.) as these last for periods no shorter than one round-trip time. However, one should keep in mind that the timing at which events occur in the model (e.g., drops or transitions between TCP modes) are only accurate up about one round-trip time. Finally, it is important to note that, although, timing is only accurate up to roughly one round-trip time, since the variations on the round-trip time typically occur at a slower time scale, the hybrid models capture quite accurately the dynamics of the round-trip time evolution. In fact, that is one of the strengths of the models proposed here, which do not assume constant round-trip time.

The remaining of this paper is organized as follows. In the next section we introduce an hybrid model for TCP congestion control. This model builds upon the one originally proposed in [17]. The two most significant new features are the explicit modeling of slow-start and fast-recovery. In Section 3 we carry out a formal analysis of the system. To achieve this we first introduce a time-normalization that renders the continuous dynamics of the hybrid model linear. We then proceed to derive several transient and steady-state properties of the system. The results obtained are compared with packet-level simulations. Section 4 contains some final conclusions and direction for future research.

# 2 Hybrid modeling of TCP

In this paper, we consider Reno congestion control. We provide next a simplified description of this algorithm that is sufficient for the purposes of this paper and refer the reader to [18, 19, 20] for a more detailed description.

Associated with each TCP flow there exists a congestion controller that possesses an internal state known as the *window size*. We denote by $w_f$, $f \in \{1, 2, \dots, n\}$, the window size of the congestion controller associated with the $f$th flow. The window size determines the maximum number of unacknowledged packets for the flow. In essence, the window size $w_f$ determines the rate $r_f$ at which the $f$th congestion controller sends packets. The relationship between $w_f$ and $r_f$ depends on the *round-trip time*, denoted by $RTT$, which is the time interval measured from the moment a packet is sent until an acknowledgment for that packet is received. Since $w_f$ packets can be sent during one round-trip time we have

$$r_f = \frac{w_f}{RTT}, \qquad\qquad f \in \{1, 2, \dots, n\}.$$

The round-trip time is given by

$$RTT(t) = T^\ell + \frac{q(t)}{B},$$

where $T^\ell$ denotes the *propagation time* (together with any fixed component of the service time) and $q(t)$ is the size of the output queue of node $\mathrm{n}_1$ at time $t$. For simplicity, we assume here that the bandwidth $B$ is measured in packets per second.

The algorithm used by TCP to update the window size $w_f$ is as follows: The congestion controller initially starts in the *slow-start mode*, where the window size $w_f$ doubles every round-trip time, leading approximately to

$$\dot{w}_f = \frac{\log 2}{RTT} w_f, \qquad\qquad r_f = \frac{w_f}{RTT}.$$

This is known as *multiplicative increase*. When a drop occurs, the congestion controller transitions to the *fast-recovery mode* and typically stays in this mode for one round-trip time[1]. Although the protocol specifies that $w_f$ actually varies during this mode, the average sending rate turns out to be approximately

$$r_f(t) = \frac{w_f(\tau)}{2RTT(t)},$$

where $\tau$ is the instant when the controller entered this mode. We can therefore assume that during the *fast recovery mode* $w_f$ remains constant and we have

$$\dot{w}_f = 0, \qquad\qquad r_f = \frac{w_f}{2RTT}.$$

Once the congestion controller leaves the *fast recovery mode*, it transitions to the *congestion avoidance mode*, where the window size increases by one packet every round-trip time, leading approximately to

$$\dot{w}_f = \frac{1}{RTT}, \qquad\qquad r_f = \frac{w_f}{RTT}.$$

This is known as *additive increase*. Once a drop occurs the congestion controller transitions again to the *fast recovery mode* mentioned above. For simplicity, we ignored the existence of timeouts. These could have been considered, as shown in [21].

[1]This is consistent with Reno, New Reno, and SACK for the case of a single drop. If multiple drops occur this particular model is only consistent with SACK [6].

The output queue at node $n_1$ receives a total of $r := \sum_f r_f$ packets per second and is able to send $B$ packets to the link in the same period. The difference between these two quantities determines the evolution of $q(t)$. In particular,

$$\dot{q} = \begin{cases} 0 & q = 0, r < B \quad \text{or} \quad q = q_{\max}, r > B \\ r - B & \text{otherwise} \end{cases} \tag{2}$$

The first branch in (2) takes into account that the queue size cannot become negative nor should it exceed the *maximum queue size* $q_{\max}$. When $q(t)$ reaches $q_{\max}$ drops occur. These will be detected by the congestion controllers and lead to a transition between modes. Since a drop will only be detected after one round-trip time, the rate of incoming packets will not change for a period of length $RTT$ and multiple drops are expected. It turns out that under drop-tail queuing policy exactly one drop per flow will occur in most operating conditions [5]. To understand why, we must recall that during the period in which there are no drops, the window size of each flow will increase by one in every round-trip time. When the acknowledgment that triggers this increase arrives, the congestion controller will attempt to send two packets *back-to-back*. The first packet is sent because the acknowledgment that just arrived decreased the number of unacknowledged packets and therefore a new packet can be sent. The second packet is sent because the window size just increased, allowing the controller to have an extra unacknowledged packet. However, at this point there is a very fragile balance between the number of packets that are getting in and out of the queue, so two packets will not fit in the queue and the second packet is dropped.

The system described above can be modeled by the hybrid system in Figure 2. In this model, we split each state of the congestion controller into two discrete states for the hybrid system. One corresponding the queue not being full and another to the queue being full. Each ellipse in this figure corresponds to a discrete mode and the continuous state of the hybrid system consists of the queue size $q$, the window sizes $w_f$ and a timing variable $t_{\text{tim}}$ used to enforce that the system remains in the *queue-full* and *fast-recovery* modes for $RTT$ seconds. The differential equations for the continuous state in each discrete mode are shown inside the corresponding ellipse. The arrows in the figure represent discrete transitions between modes. These transitions are labeled with their enabling conditions (followed by "?") and any necessary reset of the continuous state that must take place when the transition occurs (with the corresponding assignments denoted by := ). We assume here that a jump always occurs when the transition condition is enabled. The transition on the top-left entering the *slow-start/queue-not-full* represents the system's initialization. This model is consistent with most of the hybrid system frameworks proposed in the literature (cf. [22] and references therein). The transitions into the *fast-recovery/queue-full* state only occur when halving the window sizes is not sufficient to lead to a decrease in queue length, in which case multiple drops will occur. We assume here that such situation does not occur. We also assume that the queue never empties. Later we will actually establish conditions under which these assumptions hold.

Figure 3 shows a MATLAB/Simulink simulation of the hybrid model of a dumbbell topology for a single flow with TCP-SACK congestion control with rate-halving. For comparison purposes, Figure 4 shows a MATLAB/Simulink simulation of a hybrid model that ignores slow start and fast recovery [17].

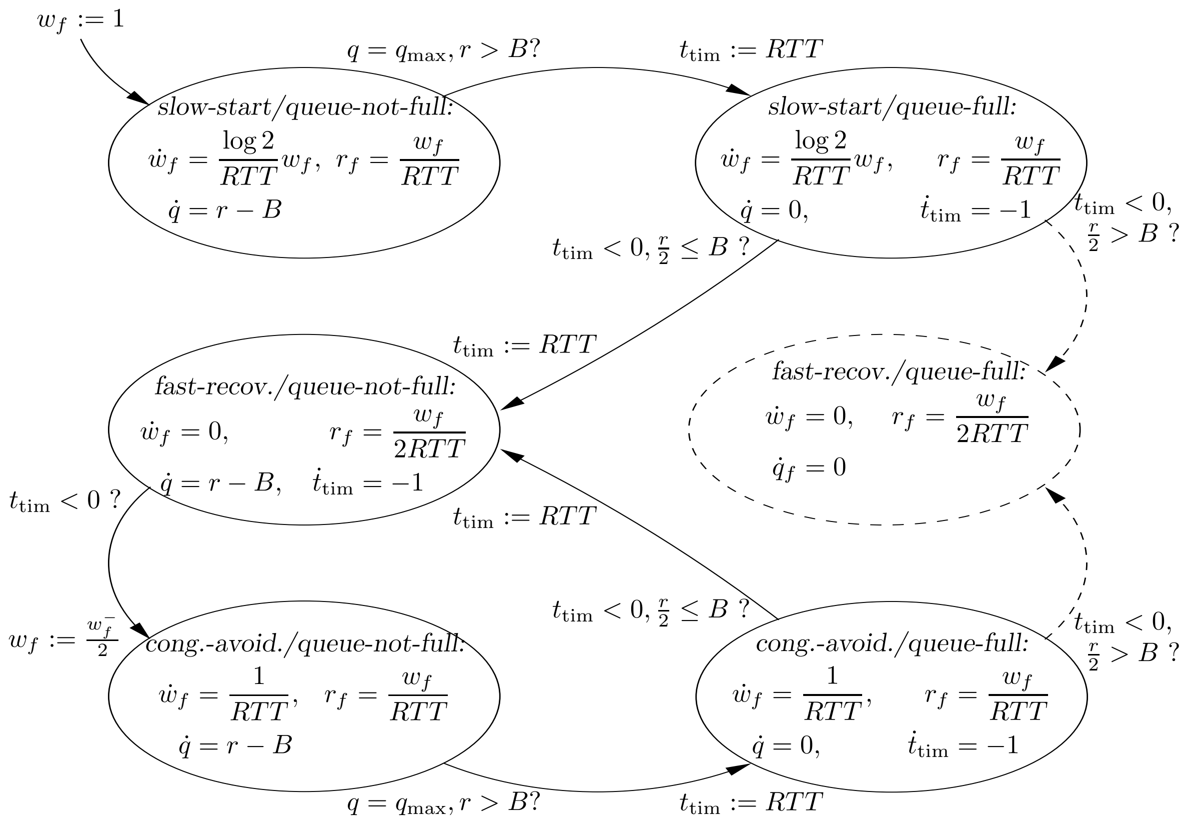


Figure 2: Hybrid model for a dumbbell network with TCP-SACK congestion control. In this figure $q := \sum_{f \in \mathcal{F}} q_f$, $r := \sum_{f \in \mathcal{F}} r_f$, and $RTT = T^{\ell} + \frac{q}{B}$.

# 3 Hybrid system analysis

We proceed now to analyze the evolution of the hybrid system. Our analysis will show that the window sizes converge to a periodic regimen, regardless of their values at the end of the slow-start period. Because we are considering the variations of the round-trip times caused by varying queuing delays, this regimen is more complex (but also closer to reality) than the simple saw-tooth wave form that is often used to characterize the

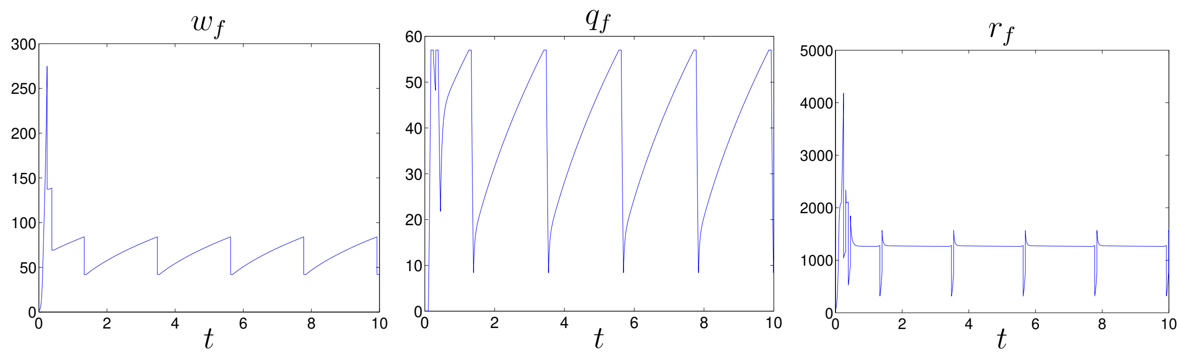


Figure 3: MATLAB/Simulink simulation of the hybrid model in Figure 2. The parameters for this simulation are $n = 1$, $B = \frac{10^7}{8 \times 1000}$, $T_p = .02$, and $q_{\max} = 57$. From the simulation we obtained $\overline{RTT} = .051$, $p = .0003728$, $T = 1250.4$.

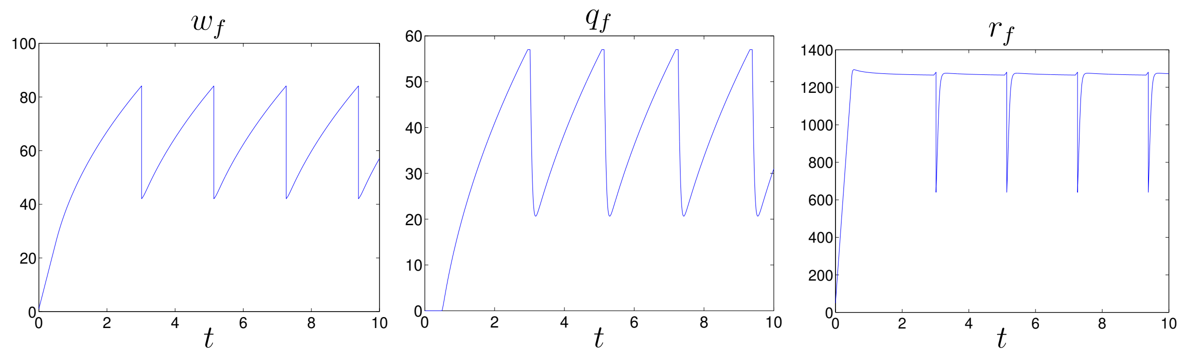


Figure 4: MATLAB/Simulink simulation of a hybrid model that ignores slow start and fast recovery [17]. Same parameters as in Figure 3.

steady-state behavior of TCP.

### 3.1 Time normalization

The dynamics for the model in Figure 2 are nonlinear because of $RTT$'s dependence on $q$. However, it is possible to make them linear by normalizing the time variable. To this effect we introduce a new time variable $\tau$, called *normalized time*[2], defined by

$$\frac{\mathrm{d}\tau}{\mathrm{d}t} = \frac{1}{RTT} = \frac{B}{BT_p + q}, \qquad \tau(0) = 0. \tag{3}$$

This means that an interval with duration $d\tau$ in the variable $\tau$ corresponds to an interval of duration $dt = RTT d\tau$ in the variable $t$. We can think of $\tau$ as a time variable normalized so that one unit of $\tau$ corresponds to one round-trip time. Figure 5 show the dynamics of the hybrid model in normalized time. In these figures, $'$ denotes the derivative $\frac{\mathrm{d}}{\mathrm{d}\tau}$ with respect to the normalized time $\tau$.

It is interesting to note that the equation for $q$ in the *queue-not-full* modes are stable. This is an important property of window-based congestion control, as opposed to other congestion control mechanisms that adapt the packets sending rates directly, instead of indirectly through the window size. This is a manifestation of the well known property that window-based mechanisms automatically adjust their sending rates to variations in the round-trip time.

The time-normalized models derived above will be used in the next section to perform a formal analysis of TCP congestion control. It should be emphasized that these linearizations are exact and do not carry any approximation.

### 3.2 Transient behavior

We are interested here in characterizing the short-term evolution—also known as the *transient behavior*—of the window sizes until the periodic regimen is reached. To this effect, let us denote by $\{\tau_k : \tau_k \leq \tau_{k+1}, k \geq 1\}$ the set of normalized times at which the system enters the *fast-recovery/queue-not-full* state. Solving the (linear) differential

[2]Formally, there is a bijective function $f$ that maps normalized time $\tau$ into real time $t$. This function is actually defined by (3). With some abuse of notation, when we write $q(\tau)$ for some normalized time $\tau$, we really mean $q(f(\tau))$. Similar notation is used for the remaining time-dependent variables.

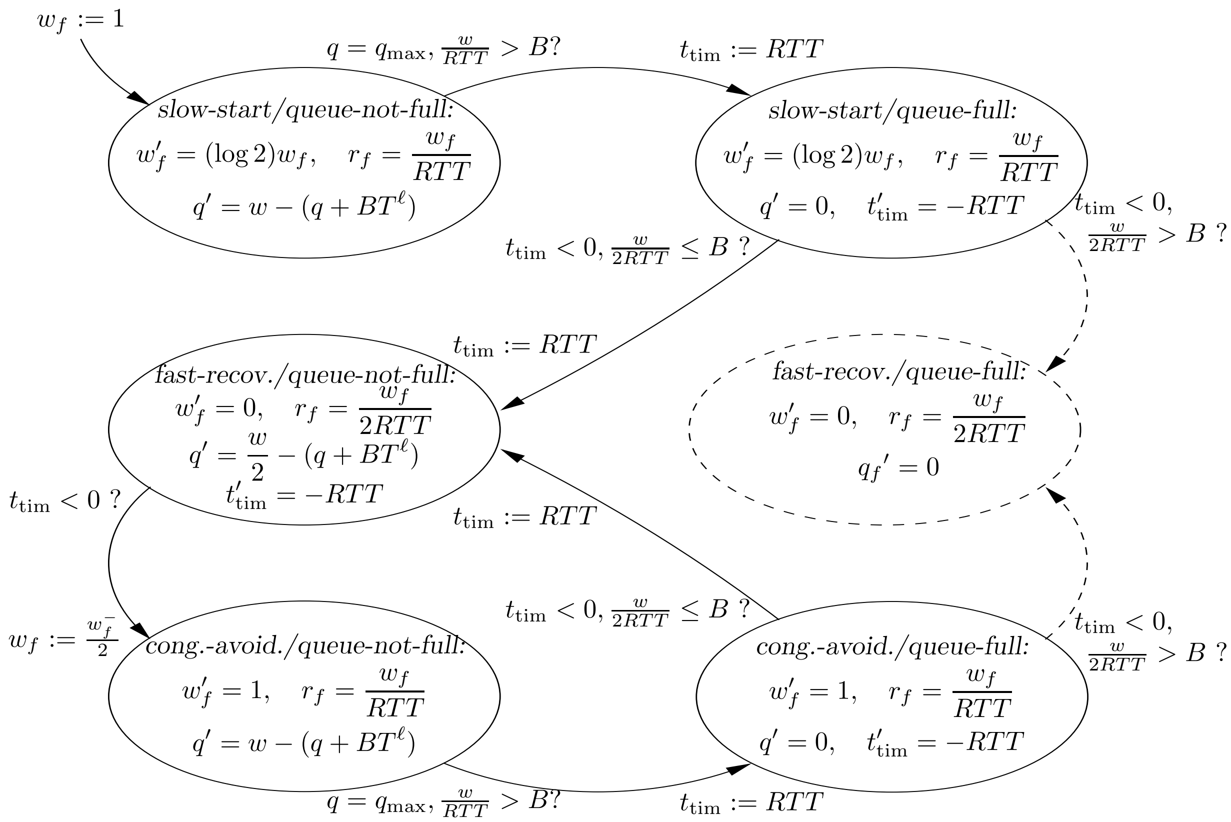


Figure 5: Hybrid model in normalized time for a dumbbell network with TCP-SACK congestion control. In this figure $q := \sum_{f \in \mathcal{F}} q_f$, $w := \sum_{f \in \mathcal{F}} w_f$, and $RTT = T^\ell + \frac{q}{B}$.

equations on each discrete state we conclude that in each interval $[\tau_k, \tau_{k+1}]$, we have the following trajectory[3]:

1. For $\tau \in [\tau_k, \tau_k + \delta_k)$ the system stays in the *fast-recovery/queue-not-full* state with

$$\begin{aligned} w_f(\tau) &= w_f(\tau_k), \\ q(\tau) &= e^{-(t-\tau_k)}\pi^\ell - BT^\ell - (1 - e^{-(t-\tau_k)})w(\tau_k)/2, \\ t_{\text{tim}} &= \frac{e^{-(t-\tau_k)}\pi^\ell + (1 - e^{-(t-\tau_k)} - t + \tau_k)w(\tau_k)/2}{B}, \end{aligned}$$

   where $\pi^\ell := q_{\max} + BT^\ell$ denotes the "size of the pipe" and $\delta_k$ is defined by the equation $t_{\text{tim}}(\tau_k + \delta_k) = 0$, and is given by

$$\delta_k := 1 + f\Big(\frac{\pi^\ell - w(\tau_k)/2}{ew(\tau_k)/2}\Big),$$

   where $f(z)$, $z > e^{-1}$ is the unique solution $w$ to the equation $we^w = z$.

2. For $\tau \in [\tau_k + \delta_k, \tau_{k+1}]$ the system is in the *congestion-avoidance/queue-not-full* state until $\tau_{k+1} - 1$ and from then on in *congestion-avoidance/queue-full* state, with

$$w_f(\tau) = \frac{w_f(\tau_k)}{2} + \tau - \tau_k - \delta_k$$

[3]To derive the following equations we used Mathematica to solve the differential equations on each discrete state.

$$q(\tau) = \begin{cases} e^{-(t-\tau_k-\delta_k)}\big(n + (\delta_k - 1)\frac{w_f(\tau_k)}{2}\big) + \frac{w_f(\tau_k)}{2} - BT^\ell + n(t - \tau_k - \delta_k - 1) & \tau \leq \tau_{k+1} - 1 \\ q_{\max} & \tau > \tau_{k+1} - 1 \end{cases}$$

and $\tau_{k+1}$ is defined by the equation $q(\tau_{k+1} - 1) = q_{\max}$, and is given by

$$\tau_{k+1} = \tau_k + \delta_k + 2 + \frac{\pi^\ell - w(\tau_k)/2}{an} + g\Big(\frac{w(\tau_k)}{n}, \frac{\pi^\ell}{n}\Big), \tag{4}$$

where

$$g(\bar{w}, \bar{\pi}) := f\left(-e^{-1-\bar{\pi}+\bar{w}/2}\Big(1 + f\Big(\frac{\bar{\pi} - \bar{w}/2}{e\bar{w}/2}\Big)\frac{\bar{w}}{2}\Big)\right), \qquad \bar{\pi} > \bar{w}/2,$$

and therefore

$$w_f(\tau_{k+1}) = \frac{w_f(\tau_k)}{2} + \frac{\pi^\ell + 2n - w(\tau_k)/2}{n} + g\Big(\frac{w(\tau_k)}{n}, \frac{\pi^\ell}{n}\Big), \tag{5}$$

For (4) to be well defined we must have $\pi^\ell > w(\tau_k)/2$, this condition essentially guarantees that one drop per flow is sufficient to take the system away from the *queue-full* state. On the other hand, the minimum value of the queue is equal to

$$\frac{w(\tau_k)}{2} - BT^\ell + n\log\left(1 + f\Big(\frac{\pi^\ell - w(\tau_k)/2}{ew(\tau_k)/2}\Big)\frac{w(\tau_k)/2}{n}\right),$$

and we are assuming here that this value is positive. The two conditions above, which are needed for the validity of our analysis, hold provided that

$$2BT^\ell \leq w(\tau_k) < 2\pi^\ell = 2(q_{\max} + BT^\ell),$$

i.e., the sum of all the window sizes $w(\tau_k)$ when a drop occurs must be between twice the number of bytes $BT^\ell$ that fit in the wire and twice the size of the pipe $\pi^\ell$, otherwise the queue will get full or a multiple drops per flow must occur, respectively. These conditions hold almost always, as when a drop occurs the pipe is full by the outstanding packets and therefore $w(\tau_k)$ is approximately equal to $\pi^\ell$.

Suppose now that we define $x$ to be a vector containing $q$ and all the $w_f$, i.e.,

$$x := \begin{bmatrix} w_1 & w_2 & \cdots & w_n & q \end{bmatrix}'.$$

This vector essentially contains all the interesting components of the continuous state of the hybrid system. Since we conclude from (5) that $x(\tau_{k+1})$ is completely determined by $x(\tau_k)$, $k \geq 1$ and we can write

$$x(\tau_{k+1}) = T(x(\tau_k)), \qquad k \geq 1,$$

where $T$ is an appropriately defined operator from the set of $n+1$-dimensional vectors with nonnegative entries into itself. It turn out that under appropriate conditions on the window sizes, there exists a norm $\|\cdot\|_\epsilon$ such that $T$ is a contraction and, in particular, that

$$\|T(x) - T(y)\|_\epsilon \leq \frac{1}{2}\|x - y\|_\epsilon, \qquad \forall x, y \tag{6}$$

(cf. Appendix for details). Because of the Contraction Mapping Theorem [23, p. 126], we then conclude that $x(\tau_k)$ converges as $k \to \infty$ to the unique fixed point of $T$.

To determine the fixed point $x$ of $T$, note that from (5) one concludes that

$$\sum_{f \in \mathcal{F}} T_f(x) = \pi^\ell + 2n + ng\Big(\frac{\bar{x}}{n}, \frac{\pi^\ell}{n}\Big),$$

where $T_f(x)$ denotes the $f$th component of the vector $T(x)$ and $\bar{x} := \sum_{f \in \mathcal{F}} x_f$, with $x_f$ the $f$th component of $x$. Therefore $T(x) = x$ implies that

$$\bar{x} = \pi^\ell + 2n + ng\Big(\frac{\bar{x}}{n}, \frac{\pi^\ell}{n}\Big). \tag{7}$$

Once $\bar{x}$ is found using the above equation, from (5) we conclude that $T_f(x) = x_f$ implies that

$$\frac{x_f}{2} + \frac{\pi^\ell + 2n - \bar{x}/2}{n} + g\Big(\frac{\bar{x}}{n}, \frac{\pi^\ell}{n}\Big) = x_f$$

From this and (7) one concludes that $x_f = \frac{\bar{x}}{n}$. The following was proved:

**Theorem 1.** *Let $\{t_k : t_k \leq t_{k+1}, k \geq 1\}$ be the set of times at which the system enters the fast-recovery/queue-not-full state. Assuming that*

$$2BT^\ell \leq w(t_1) < 1.8\pi^\ell = 1.8(q_{\max} + BT^\ell), \qquad\qquad \pi^\ell \geq 2n,$$

*then all the $w_f(t_k)$, $f \in \mathcal{F}$ converge exponentially fast to $\frac{\bar{x}}{n}$ as $k \to \infty$, with $\bar{x}$ the solution to* (7)*. The convergence is as fast as $\left(\frac{1}{2}\right)^k$.*

The requirement on $w(\tau_1)$ guarantees that one drop per flow is sufficient to take the system away from the *queue-full* modes (right inequality) and that the queue never empties (left inequality). Since when a drop occurs the pipe is filled by the unacknowledged packets, the $w(\tau_k)$ are always approximately equal to $\pi^\ell$ so the right inequality generally holds. As for the left inequality, it holds as long as $q_{\max} \geq BT^\ell$, i.e., as long as the queueing capacity at the bottleneck link exceeds the "wire capacity." This condition should be taken into consideration when allocating buffer capacity as its violation will lead to empty queues and therefore underutilization of the available bandwidth.

## 3.3 Steady-state behavior

In the previous section we established that the window sizes converge to a periodic regimen, also known as the *steady-state* regimen. We proceed now to analyze the system when it operates under this regimen. Among other things, we will show that a relationship between average throughput, average drop rate (i.e., the percentage of dropped packets), and average round-trip time such as (1) can also be derived from our model.

For any reasonable sized pipe (e.g., larger than 5 packets) $\pi^\ell + 2n \gg ng\Big(\frac{\bar{x}}{n}, \frac{\pi^\ell}{n}\Big)$ and, because of (7), we have $\bar{x} \approx \pi^\ell + 2n$ and therefore $w(\tau_k) \to \pi^\ell + 2n$, $w_f(\tau_k) \to \frac{\pi^\ell}{n} + 2$ as $k \to \infty$. Moreover, when the steady-state is reached,

$$\tau_{k+1} \approx \tau_k + \delta_k + 1 + \frac{\pi^\ell}{2n} \tag{8}$$

$$w_f(\tau) \approx \begin{cases} \frac{\pi^\ell}{2n} + 1, & \tau \in [\tau_k, \tau_k + \delta) \\ \frac{\pi^\ell}{2n} + \tau - \tau_k - \delta + 1 & \tau \in [\tau_k + \delta, \tau_{k+1}] \end{cases} \tag{9}$$

$$q(\tau) \approx \begin{cases} e^{-(t-\tau_k)}\pi^\ell - BT^\ell - (1 - e^{-(t-\tau_k)})\frac{\pi^\ell+2n}{2}, & \tau \in [\tau_k, \tau_k + \delta) \\ e^{-(t-\tau_k-\delta)}\big(an + (\delta - 1)\frac{\pi^\ell+2n}{2n}\big) + \frac{\pi^\ell+2n}{2n} - BT^\ell + n(t - \tau_k - \delta - 1) & \tau \in [\tau_k + \delta, \tau_{k+1} - 1] \\ q_{\max} & \tau \in (\tau_{k+1} - 1, \tau_{k+1}] \end{cases} \tag{10}$$

with $\delta := 1 + f\left(\frac{\pi^\ell - 2n}{e(\pi^\ell+2n)}\right) \approx 1$.

Suppose then that the steady-state has been reached and let us consider an interval $[t_k, t_{k+1}]$ between two consecutive time instants at which the system enters the *fast-recovery/queue-not-full* state. The total number of packets $N_k$ sent during the interval $[t_k, t_{k+1}]$ can then be computed by

$$N_k := \int_{t_k}^{t_{k+1}} r(t)dt = \int_{\tau_k}^{\tau_{k+1}} r(\tau)RTT(\tau)d\tau = \int_{\tau_k}^{\tau_{k+1}} w(\tau)d\tau \approx \frac{3n(\pi^\ell/n + 2)(\pi^\ell/n + 14/3)}{8} \tag{11}$$

We used here the change of integration variable defined by (3) to work with quantities in normalized time. The integrals in (11) and (13) below were computed symbolically from (8)–(10), using Wolfram Research's Mathematica software. Since $n$ drops occur in the interval $[t_k, t_{k+1}]$, *the average drop rate* $p$ is then equal to

$$p := \frac{n}{N_k} \approx \frac{8}{3(\pi^\ell/n + 2)(\pi^\ell/n + 14/3)}. \tag{12}$$

Another quantity of interest is the average round-trip time $\overline{RTT}$. We consider here a packet-average, rather than a time-average, because the former is the one usually measured in real networks. This distinction is important since the sending rate $r$ is not constant. In fact, when the sending rate is higher, the queue is more likely to be full and the round-trip time is larger. This results in the packet-average being larger than the time-average. The *packet-average round-trip time* can then be computed as[4]

$$\begin{aligned} \overline{RTT} &:= \frac{\int_{t_k}^{t_{k+1}} r(t)RTT(t)dt}{N_k} = \frac{\int_{\tau_k}^{\tau_{k+1}} w(\tau)RTT(\tau)d\tau}{N_k} \\ &= \frac{1}{T}\frac{7e(\pi^\ell/n)^3 + 3(19e - 4)(\pi^\ell/n)^2 + 4(17e + 12)}{9e(\pi^\ell/n + 2)(\pi^\ell/n + 14/3)}, \end{aligned} \tag{13}$$

where $T := \frac{B}{n}$ is the average throughput of each flow. We recall that, because the queue never empties, the total throughput is precisely the bandwidth $B$ of the bottleneck link.

It is interesting to note that the average drop rate $p$ can provide an estimate for per-flow pipe size $\pi^\ell/n$. In particular, we conclude from (12) that

$$\frac{\pi^\ell}{n} \approx \hat{\pi}_f^\ell := \sqrt{\frac{8(3 + 2p)}{9p}} - \frac{10}{3} \tag{14}$$

[4]Here, we dropped a few terms on $e^{-1-\frac{\pi^\ell}{2n}}$ that was dominated by the ones kept.

This, in turn, can be used together with (13) to estimate the average throughput $T$. In fact, from (13) and (14) we conclude that

$$T \approx \frac{1}{\overline{RTT}} \frac{7e(\hat{\pi}^\ell_f)^3 + 3(19e-4)(\hat{\pi}^\ell_f)^2 + 4(17e+12)}{9e(\hat{\pi}^\ell_f + 2)(\hat{\pi}^\ell_f + 14/3)}.$$

The following theorem summarizes the results above:

**Theorem 2.** *Under the hypothesis of Theorem 1, the average drop rate $p$, the packet average round-trip time $\overline{RTT}$, and the average throughput $T$ of each flow are approximately given by*

$$p \approx \frac{8a}{3(\pi^\ell/n + 2a)(\pi^\ell/n + 14a/3)} \tag{15}$$

$$\overline{RTT} \approx \frac{n}{B} \frac{7e(\pi^\ell/n)^3 + 3(19e-4)(\pi^\ell/n)^2 + 4(17e+12)}{9e(\pi^\ell/n + 2)(\pi^\ell/n + 14/3)} \tag{16}$$

$$T \approx \frac{1}{\overline{RTT}} \frac{7e(\hat{\pi}^\ell_f)^3 + 3(19e-4)(\hat{\pi}^\ell_f)^2 + 4(17e+12)}{9e(\hat{\pi}^\ell_f + 2)(\hat{\pi}^\ell_f + 14/3)}., \tag{17}$$

*where $\hat{\pi}^\ell_f$ is an estimate of the per-flow pipe size $\pi^\ell/n$ obtained from*(14)*.*

To verify the formulas in Theorem 2, we simulated the dumbbell topology of Figure 1, using the `ns-2` network simulator [11]. Figure 6 summarizes the results obtained for a network with the following parameters: $B = \frac{10^7 \text{ bits/sec}}{8 \text{ bits/char} \times 1000 \text{ char/packet}} = 1250$ packets/sec, $T_p = .04$ sec, $q_{\max} = 250$ packets. As seen in the figure, the theoretical predictions given by (15)–(17) match the simulation results quite accurately. Note the comparison in the rightmost plot of the theoretical prediction obtained by (17) and that obtained using the standard formula (1). One can see that the model derived here is valid over a wider range of traffic conditions (almost one order of magnitude larger).

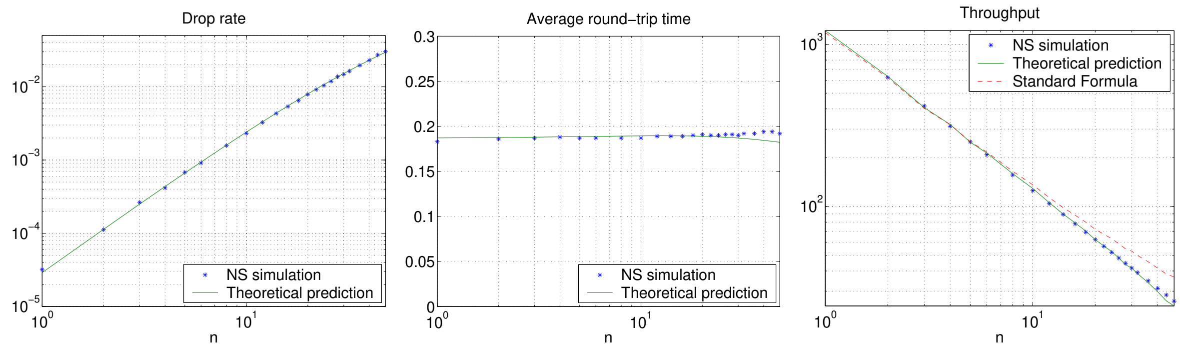


Figure 6: Comparison between the predictions obtained from the hybrid model and the results from `ns-2` simulations.

# 4 Conclusions

We presented an hybrid model for the dynamics of multiple TCP flows that share a single common bottleneck link. The main contributions of this paper were a characterization of the transient behavior of the flows, predicting their experimentally observed exponential synchronization as well as a characterization of the steady-state regimen that is significantly more accurate than existing ones, especially under heavy traffic. The transient analysis of TCP also provides rules for network provisioning. For example, we showed that the queueing capacity at the bottleneck should not be smaller than the "wire-capacity." We are now in the process of generalizing this type of model to other network topologies and other types of congestion control. Another direction we are exploring is the application of the hybrid models derived here to detect abnormalities in TCP traffic flows. This has important applications in network security.

# Appendix

*Proof of equation* (6). Consider the norm

$$\|x\|_\epsilon := \Big|\sum_{f\in\mathcal{F}} x_f\Big| + \epsilon \sum_{f\in\mathcal{F}} |x_f|,$$

defined on the set

$$\mathcal{X} := \Big\{x := \begin{bmatrix} x_1 & x_2 & \cdots & x_{n+1}\end{bmatrix}' : x_f \geq 0,\ \sum_{f\in\mathcal{F}} x_f \leq L\Big\}, \qquad L < 2\pi^\ell,$$

where $\epsilon$ is a positive constant. For any two $x, y \in \mathcal{X}$, the $f$th component of $T(x) - T(y)$ is given by

$$T_f(x) - T_f(y) = \begin{cases} \dfrac{x_f - y_f}{2} - \dfrac{\bar{x} - \bar{y}}{2n} + g\Big(\dfrac{\bar{x}}{n}, \dfrac{\pi^\ell}{n}\Big) - g\Big(\dfrac{\bar{y}}{n}, \dfrac{\pi^\ell}{n}\Big) & f \in \mathcal{F} \\ 0 & f = n+1 \end{cases}$$

where $\bar{x} := \sum_{f'\in\mathcal{F}} x_{f'}$, $\bar{y} := \sum_{f'\in\mathcal{F}} y_{f'}$, and therefore

$$\begin{aligned}
\|T(x) - T(y)\|_\epsilon &= \Big| g\Big(\frac{\bar{x}}{n}, \frac{\pi^\ell}{n}\Big) - g\Big(\frac{\bar{y}}{n}, \frac{\pi^\ell}{n}\Big) \Big| \\
&\quad + \epsilon \sum_{f\in\mathcal{F}} \Big| \frac{x_f - y_f}{2} - \frac{\bar{x} - \bar{y}}{2n} + g\Big(\frac{\bar{x}}{n}, \frac{\pi^\ell}{n}\Big) - g\Big(\frac{\bar{y}}{n}, \frac{\pi^\ell}{n}\Big) \Big| \\
&\leq (1+\epsilon) \Big| g\Big(\frac{\bar{x}}{n}, \frac{\pi^\ell}{n}\Big) - g\Big(\frac{\bar{y}}{n}, \frac{\pi^\ell}{n}\Big) \Big| + \frac{\epsilon}{2} \sum_{f\in\mathcal{F}} |x_f - y_f| + \frac{\epsilon|\bar{x} - \bar{y}|}{2} \\
&= (1+\epsilon) \Big| g\Big(\frac{\bar{x}}{n}, \frac{\pi^\ell}{n}\Big) - g\Big(\frac{\bar{y}}{n}, \frac{\pi^\ell}{n}\Big) \Big| + \frac{1}{2}\|x - y\|_\epsilon - \frac{1-\epsilon}{2}|\bar{x} - \bar{y}| \\
&\leq \frac{1}{2}\|x - y\|_\epsilon,
\end{aligned}$$

provided that

$$(1+\epsilon) \Big| g\Big(\frac{s_1}{n}, \frac{\pi^\ell}{n}\Big) - g\Big(\frac{s_2}{n}, \frac{\pi^\ell}{n}\Big) \Big| \leq \frac{1-\epsilon}{2}|s_1 - s_2|, \qquad \forall s_1, s_2 \in [0, L].$$

We therefore conclude that one can find a norm under which $T$ is a contraction on $\mathcal{X}$, as long as

$$\Big| g\Big(\frac{s_1}{n}, \frac{\pi^\ell}{n}\Big) - g\Big(\frac{s_2}{n}, \frac{\pi^\ell}{n}\Big) \Big| < \frac{1}{2}|s_1 - s_2|, \qquad \forall s_1, s_2 \in [0, L],$$

or, more conservatively, as long as

$$\left|\frac{\mathrm{d}g}{\mathrm{d}\bar{w}}\Big(\frac{L}{an},\frac{\pi^\ell}{n}\Big)\right| < \frac{n}{2}$$

Here, we used the fact that the norm of the derivative of $g(\bar{w},\bar{\pi})$ increases with $\bar{w}$. It turns out that this condition holds for every $n \geq 1$, e.g., as long as $\pi^\ell \geq 2an$ (i.e., at least 2 packets per flow can fit in the pipe) and $L$ is above $\pi^\ell$ by at most 80% (i.e., the number of outstanding packets does not exceed the size of the pipe by more than 80%). ■

# References


[1] J. Padhye, J. Kurose, D. Towsley, and R. Koodli, "A TCP-friendly rate adjustment protocol for continuous media flows over best effort networks," Tech. Rep. TR 89-04, UMASS-CMPSCI, 1998.

[2] S. Floyd, M. Handley, and J. Padhye, "A comparison of equation-based and AIMD congestion control." Available at `http://www.aciri.org/tfrc/`, May 2000.

[3] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *Proc. of SIGCOMM*, pp. 43–56, Aug. 2000.

[4] D. Bansal and H. Balakrishnan, "Binomial congestion control algorithms," in *Proc. of INFOCOMM*, pp. 631–640, Apr. 2001.

[5] L. Zhang, S. Shenker, and D. D. Clark, "Observations on the dynamics of a congestion control algorithm: The effects of two-way traffic," in *Proc. of SIGCOMM*, Sept. 1991.

[6] K. Fall and S. Floyd, "Simulation-based comparisons of Tahoe Reno and SACK TCP," *ACM Comput. Comm. Review*, vol. 27, pp. 5–21, July 1996.

[7] T. Ott, J. H. B. Kemperman, and M. Mathis, "Window size behavior in TCP/IP with constant loss probability," in *Proc. of the DIMACS Workshop on Performance of Realtime Applications on the Internet*, Nov. 1996.

[8] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The macroscopic behavior of the TCP congestion avoidance algorithm," *ACM Comput. Comm. Review*, vol. 27, July 1997.

[9] R. Morris, *Scalable TCP Congestion Control*. PhD thesis, Harvard University, Cambridge, MA, Jan. 1999.

[10] E. Altman, C. Barakat, and E. Laborde, "Fairness analysis of TCP/TP," in *Proc. of the 39th Conf. on Decision and Contr.*, pp. 61–66, Dec. 2000.

[11] The VINT Project, a collaboratoin between researchers at UC Berkeley, LBL, USC/ISI, and Xerox PARC, *The ns Manual (formerly ns Notes and Documentation)*, Oct. 2000. Available at http://www.isi.edu/nsnam/ns/ns-documentation.html.

[12] J. Mahdavi and S. Floyd, "TCP-friendly unicast rate-based flow control." Technical note sent to the end2end-interest mailing list, Jan. 1997.

[13] T. V. Lakshman, U. Madhow, and B. Suter, "Window-based error recovery and flow control with a slow acknowledgment channel: A study of TCP/IP performance," in *Proc. of INFOCOMM*, Apr. 1997.

[14] V. Misra, W. Gong, and D. Towsley, "Stochastic differential equation modeling and analysis of TCP-windowsize behavior," in *In Proceedings of PERFORMANCE99*, (Istanbul, Turkey), 1999.

[15] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: a simple model and its empirical validation," in *Proc. of SIGCOMM*, Sept. 1998.

[16] V. Misra, W. Gong, and D. Towsley, "Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED," in *Proc. of SIGCOMM*, Sept. 2000.

[17] J. P. Hespanha, S. Bohacek, K. Obraczka, and J. Lee, "Hybrid modeling of TCP congestion control," in *Hybrid Systems: Computation and Control* (M. D. D. Benedetto and A. Sangiovanni-Vincentelli, eds.), no. 2034 in Lecture Notes in Computer Science, pp. 291–304, Berlin: Springer-Verlag, Mar. 2001.

[18] V. Jacobson, "Congestion avoidance and control," in *Proc. of SIGCOMM*, vol. 18.4, pp. 314–329, Aug. 1988.

[19] V. Jacobson, "Modified TCP congestion avoidance algorithm." Posted on end2end-interest mailing list, Apr. 1990. Available at ftp://ftp.ee.lbl.gov/email/vanj.90apr30.txt.

[20] M. Allman, V. Paxson, and W. Stevens, "TCP congestion control," *RFC 2581*, p. 13, Apr. 1999.

[21] S. Bohacek, J. P. Hespanha, J. Lee, and K. Obraczka, "A hybrid systems framework for TCP congestion control: A theoretical model and its simulation-based validation (extended version)," tech. rep., University of Southern California, Los Angeles, CA, July 2001. Available at *http://www-rcf.usc.edu/˜hespanha/techreps.html*.

[22] A. van der Schaft, *An Introduction to Hybrid Dynamical Systems*. No. 251 in Lecture Notes in Control and Information Sciences, London: Springer-Verlag, 2000.

[23] A. W. Naylor and G. R. Sell, *Linear Operator Theory in Engineering and Science*. No. 40 in Applied Mathematical Sciences, New York: Springer-Verlag, 1982.