Hybrid Modeling of TCP Congestion Control*

João P. Hespanha† hespanha@usc.edu
Stephan Bohacek‡ bohacek@math.usc.edu
Katia Obraczka§ katia@isi.edu
Junsoo Lee§ junsoole@scf.usc.edu

†*Dept. Electrical Engineering–Systems, Univ. of Southern California*
3740 McClintock Ave., EEB 318, Los Angeles, CA 90089-2563
phone: (213) 740-9137, fax: (213) 821-1109

‡*Department of Mathematics, Univ. of Southern California*
1042 West 36th Place, DRB 155, Los Angeles, California 90089-1113

§*Information Science Institute, Univ. of Southern California*
4676 Admiralty Way, Marina del Rey, CA 90292

November 28, 2000

Abstract

In this paper we propose a hybrid model for TCP congestion control mechanism operating under drop-tail queuing policy. Using this model we confirmed the standard formula $T := \frac{\sqrt{2/3}}{\overline{RTT}\sqrt{p}}$ used by TCP-friendly congestion control mechanisms, which relates the average packet drop rate p, the average round-trip time $\overline{RTT}$, and the average throughput T. The hybrid model also allows us to understand the transient behavior and theoretically predict the flow synchronization phenomena that has been observed in simulations and in real networks but, to the best of our knowledge, has not been theoretically justified. This model can also be used detect abnormalities in TCP traffic flows, which has important applications in network security.

*This research was supported by the Defense Advanced Research Projects Agency and the Office of Naval Research.
†Corresponding author.

1 Introduction

Consider the computer network show in Figure 1. In this topology, n TCP flows are generated at a source node n_1 and are direct towards a sink node n_2. All the flows compete for the finite bandwidth B that characterizes the link ℓ that connects the nodes. This configuration is known as a *dumbbell topology* and is typically used to analyze TCP's congestion control. In more realistic networks, a path of several links (and intermediate nodes) would connect the source and destination nodes. However, to analyze congestion control mechanisms, one often ignores the existence of all the intermediate links, except for the *bottleneck link*, i.e., the link that has the smallest bandwidth. In the dumbbell topology, ℓ represents precisely this link.

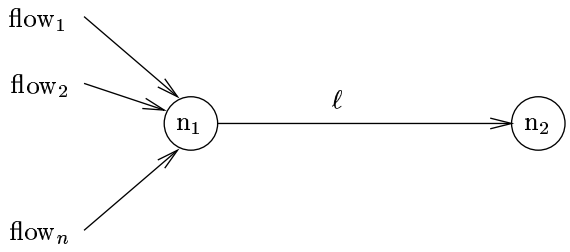

Figure 1: Dumbbell topology

The basic problem in congestion control is to determine sending rates for each of the n flows that result in an optimal utilization of the available bandwidth. The transport layer of the TCP/IP protocol stack is responsible for solving this problem and the sending rates are determined by n congestion controllers. Each congestion controller adjusts the sending rate of one particular flow, based on the number of packet drops that this flow is suffering. Packet drops occur when the sending rates of the flows are too large and the source node n_1 is unable to process all the packets received. The congestion controller becomes aware of packet drops because, each time a packet is received by the destination node, it send an *acknowledgment* packet back to the source node. When a data packet is dropped, its acknowledgment is never received and the congestion controller should take some action. The congestion control problem is nontrivial because of the following:

1. The bandwidth B associated with the link ℓ and the total number of flows n competing for this bandwidth are not known by the congestion controllers. Moreover, these parameters are likely to change over time.

2. The exchange of control information among congestion controllers and between the congestion controllers and the nodes is undesirable. This is because the control information would compete with the data for the available bandwidth.

Every computer connected to the Internet runs some version of TCP congestion control. It is therefore not surprising to find that a significant body of literature is devoted to this topic. However, many basic questions remain poorly understood. These include:

1. Does TCP congestion control work? In particular, is it able to prevent a catastrophic collapse of the network under very heavy load.

2. Is TCP congestion control fair? In particular, does it result in approximately equal throughput for all competing flows.

3. Is TCP optimal or close to optimal? This question is particularly difficult because there is no universally accepted notion of optimality. Small drops rates, small delays, approximately constant flow rates, and fast adaptation to changes in the network are certainly desirable properties. However, these criteria are self-contradictory and therefore trade-off solutions are required.

In this paper we provide a hybrid model for Reno congestion control [1]. Reno is one of the more popular versions of TCP congestion control and is generally accepted to perform well. The hybrid model developed here allowed us to shed light in some of the questions formulated above.

The model proposed provides a new derivation for the now fairly standard formula

$$T = \frac{\sqrt{3/2}}{\overline{RTT}\sqrt{p}} \tag{1}$$

that relates the average packet drop rate p, the average round-trip time $\overline{RTT}$, and the average throughput T [2, 3]. Formulas such as (1) have been used to design congestion control mechanisms that are TCP-friendly but produce more constant sending rates, making them more suitable, e.g., for streaming multimedia over the Internet [4]. Unlike previous derivations, ours considers the effect of queuing and the coupling between the competing flows.

The hybrid model presented here also predicts that the dumbbell topology in Figure 1, with drop-tail queueing at node n_1, leads to flow synchronization, i.e., the sending rates of all the flows exhibit in-phase periodic variations. This produces undesirably large variations of the round-trip time and poor utilization of the queue. This type of behavior has been observed before [5] and actually led to the development of Random Early Detection/Drop active queuing [6]. To the best of our knowledge, this is the first time that the synchronization phenomena is theoretically explained.

2 Hybrid Model for Congestion Control

In this paper, we consider Reno congestion control. We describe next a simplified version of this algorithms that is sufficient for the purposes of this paper. Each congestion controller possesses an internal state known as the *window size*. We denote by w_i, $i \in \{1, 2, \ldots, n\}$, the window size of the congestion controller associated with the ith flow. The window size determines the maximum number of unacknowledged packets for that flow. E.g., if $w_i = 3$, then the congestion controller can send 3 packets immediately, but must wait for one acknowledgment to arrive before a 4th packet can be sent. The algorithm to update the window size w_i is as follows: While no drops occur, the window size is incremented by a fixed constant $a \geq 1$ for each w_i acknowledgments received (typically $a = 1$). This is known as *additive increase*. When it is detected that a drop occurred (because an acknowledgment packet is missing) the window size is multiplied by a constant $m \in (0, 1)$ (typically $m = 1/2$). This is known as *multiplicative decrease*. We are ignoring Reno's initial adjustment of the window size known as *slow start* because it has little impact on the system after a brief initial period. The reader is referred to [1] for a detailed description of Reno congestion control.

Although the window size takes discrete values, it is convenient to regard it as a continuously varying variable. Let us call *round-trip time*, denoted by RTT, the time interval measured from the moment a packet is sent until an acknowledgment for that packet is received. As we will see below, the round-trip time is a time-varying quantity. Suppose that at some time t, the congestion controller for the ith flow sends one packet and fills its window. This means that w_i packets are now unacknowledged for. Assuming that there are no drops, after one round-trip time the acknowledgment for this packet is received, as well as the acknowledgments for the previous $w_i - 1$ packets. Since w_i acknowledgments were received, the window size must have increased by a. The following hybrid model provides a good approximation of the ith window size dynamics: While the ith flow suffers no drops we have

$$\dot{w}_i = \frac{a}{RTT}, \tag{2}$$

and if a drop is detected on this flow at time t, we have

$$w_i(t) = m\, w_i^-(t),$$

where $w_i^-(t)$ denotes the limit from below of $w_i(s)$ as $s \uparrow t$.

We proceed to determine the evolution of the round-trip time $RTT(t)$. Typically, the round trip time has two components: a fixed *propagation time* T_p that is determines by the physical length of the link ℓ and the speed of light, and a variable *service time* T_s that accounts for the time the nodes take to process the packet. The service time is usually dominated by the *queue time* T_q, i.e., the time a packet stays in the input queue of node n_1 before it is sent to the link. Denoting by $q(t)$ the size of this queue at time t, and by B the bandwidth of link ℓ in packets per second, the queuing time is given by

$$T_q(t) = \frac{q(t)}{B},$$

because $q(t)$ packets need to be transmitted (each taking $1/B$ seconds) before a new packet can also be transmitted. The round-trip time is then given by

$$RTT(t) = T_p + \frac{q(t)}{B}. \tag{3}$$

In this formula, we incorporated in T_p any fixed component of the service time.

As mentioned above, the ith flow receives w_i acknowledgment packets in one round-trip time. Therefore, in average, it sends w_i packets per round-trip time. This means that the input queue at node n_1 receives a total of $\frac{\sum_i w_i}{RTT}$ 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,

$$\dot{q} = \begin{cases} 0 & q = 0, \frac{\sum_i w_i}{RTT} < B \quad \text{or} \quad q = q_{\max}, \frac{\sum_i w_i}{RTT} > B \\ \frac{\sum_i w_i}{RTT} - B & \text{otherwise} \end{cases} \tag{4}$$

The first branch in (4) 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 some time later.

To complete our model it remains to understand how many drops occur and in which flows. As mentioned above, drops will occur whenever q reaches the maximum queue size $q_{\max}$ and the rate of incoming packets to the queue $\frac{\sum_i w_i}{RTT}$ exceeds the rate B of outgoing packets. 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, in most operating conditions, exactly one drop per flow will occur. To understand why, we must recall that in every round-trip time the window size of each flow will increase because each flow will receive as many acknowledgments as its window size. When the acknowledgment that triggers the increase of the window size by $a \geq 1$ arrives, the congestion controller will attempt to send two packets. 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. This, of course, assumes a drop-tail queuing policy. Although this behavior is essentially caused by the discreteness of the queue mechanism, we can incorporate it in our hybrid model by considering two modes for the system: One mode corresponds to the situation when the queue is not full and therefore the system evolves according to (2), (3), (4). The other mode of operation corresponds to the situation where the queue is full and one drop will occur in each flow. This mode of operation is active for RTT seconds. When the system leaves this mode all window sizes are multiplied by m because of the multiplicative decrease caused by the drops.

Figure 2 contains a graphical representation of the overall hybrid system. In this figure, each node represents one of the two discrete states: *queue-full* and *queue-not-full*. The continuous state of the hybrid system consists of the queue size q, the window sizes w_i, $i \in \{1, 2, \cdots, n\}$, and a timing variable t_T used to enforce that the system remains in the *queue-full* state for exactly RTT seconds. The differential equations for these variables in each discrete state are shown inside the corresponding nodes. The links in the figure

represent discrete transitions, which are labeled with their enabling conditions and any necessary reset of the continuous state that must take place when the transition occurs. We assume here that a jump always occurs when the transition condition is enabled. This model falls in several of the general hybrid systems frameworks proposed in the literature [7, 8, 9, 10, 11, 12, 13, 14, 15]. For simplicity we assume here that the

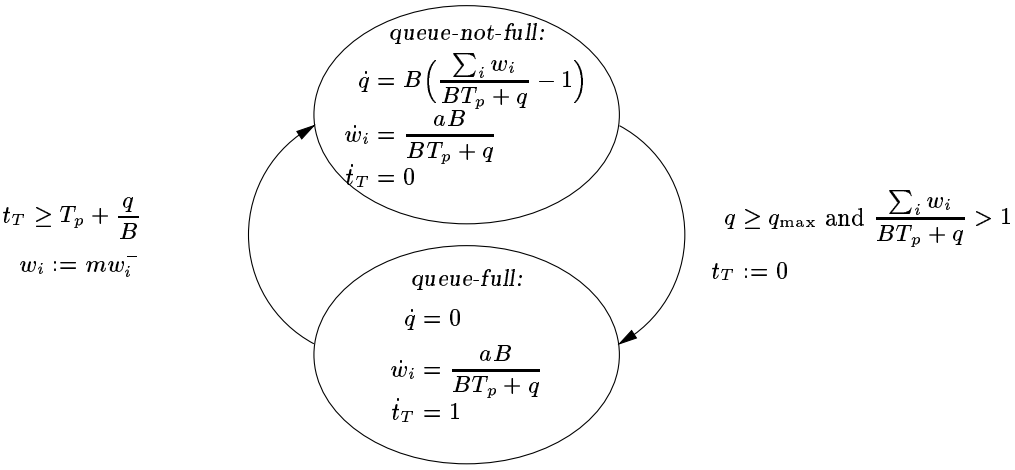

Figure 2: Hybrid model for Reno congestion control

queue size q never reaches zero.

Remark 1. For a very large number of flows, a single drop per flow may not be sufficient to produce the decrease in the window size required to make the queue size drop below $q_{\max}$ after the multiplicative decrease. In this case, the model in Figure 2 is not valid. However, for most operating conditions the above model is approximately valid as we shall see in Section 4.

3 Dynamics in normalized time

The dynamics for the hybrid system in Figure 2 are nonlinear essentially because of the dependence of RTT on q. However, it is possible to make them linear by normalizing the time variable. To this effect we introduce a new time variable τ, called the *normalized time*[1], defined by

$$\frac{\mathrm{d}t}{\mathrm{d}\tau} = RTT = T_p + \frac{q}{B}, \qquad \tau(0) = 0. \tag{5}$$

This means that an interval with duration $d\tau$ in the variable τ corresponds to an interval of duration $dt = RTT\, d\tau$ in the variable t. We can think of τ as a time variable normalized so that 1 unit of τ corresponds to one round-trip time. Figure 3 shows the dynamics of the hybrid system in normalized time. In this figure, $'$ denotes the derivative $\frac{\mathrm{d}}{\mathrm{d}\tau}$ with respect to the normalized time τ. In Figure 3, we also used the fact that in the *queue-full* state, $q = q_{\max}$ and therefore, waiting until t_T reaches $T_p + \frac{q_{\max}}{B}$ from zero with $t_T' = RTT = T_p + \frac{q_{\max}}{B}$, is equivalent to wait until τ_T reaches 1 from zero with $\tau_T' = 1$.

It is interesting to note that the equation that models the queue dynamics in the *queue-not-full* state is 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.

[1] Formally, there is a bijective function f that maps normalized time τ into real time t. This function is actually defined by (5). With some abuse of notation, when we write $q(\tau)$ for some normalized time τ, we really mean $q(f(\tau))$. Similar notation is used for the remaining time-dependent variables.

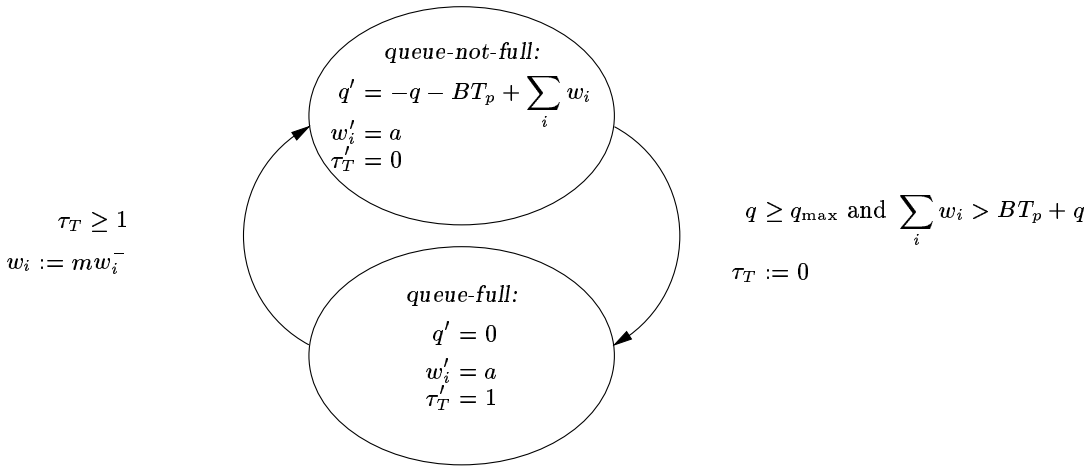

Figure 3: Hybrid model for Reno congestion control in normalized time.

Let us denote by $\{\tau_k : \tau_k \leq \tau_{k+1}, k \geq 1\}$ the set of times at which the system leaves the *queue-full* state. Because the system dynamics is essentially linear in each discrete state, it is straightforward to show [16] that

$$\tau_{k+1} - \tau_k = f^{-1}(s_k) + 1, \qquad k \geq 1, \tag{6}$$

where

$$s_k := \frac{q_{\max} + BT_p - \sum_{i=1}^n w_i(\tau_k)}{an}, \tag{7}$$

and $f : [0, \infty) \to [0, \infty)$ denotes the smooth bijection

$$x \mapsto \begin{cases} \frac{x}{1-e^{-x}} - 1 & x \neq 0 \\ 0 & x = 0 \end{cases}.$$

Denoting by t_k, $k \geq 1$, the real-time that corresponds to the normalized time τ_k, it is also straightforward to show that

$$t_{k+1} - t_k = \int_{\tau_k}^{\tau_{k+1}} \left(T_p + \frac{q(\tau)}{B} \right) d\tau = \frac{1}{B} f^{-1}(s_k) \sum_{i=1}^n w_i(\tau_k) + \frac{an}{2B} f^{-1}(s_k)^2 + T_p + \frac{q_{\max}}{B}. \tag{8}$$

We proceed to analyze the evolution of the $w_i(\tau_k)$. To this effect, suppose that the system left the *queue-full* state at some normalized time τ_k, $k \geq 1$. Since it takes $f^{-1}(s_k) + 1$ units of normalized time until the system leaves the *queue-full* state again and during this time $w_i' = a$, $i \in \{1, 2, \ldots, n\}$, we conclude that

$$w_i^-(\tau_{k+1}) = w_i(\tau_k) + af^{-1}(s_k) + a, \qquad i \in \{1, 2, \ldots, n\},$$

and therefore

$$w_i(\tau_{k+1}) = m\left(w_i(\tau_k) + af^{-1}(s_k) + a \right), \qquad i \in \{1, 2, \ldots, n\}. \tag{9}$$

From (7) and (9) we then conclude that

$$s_{k+1} = m\big(s_k - f^{-1}(s_k)\big) + \frac{1-m}{an}(q_{\max} + BT_p) - m. \tag{10}$$

It turns out that, as long as

$$q_{\max} + BT_p \geq \frac{2ma}{1-m}n,$$

the map $g : [0, \infty) \to [0, \infty)$, defined by

$$s \mapsto m\big(s - f^{-1}(s)\big) + \frac{1-m}{an}(q_{\max} + BT_p) - m,$$

is a contraction. In particular,

$$|g(s) - g(\bar{s})| = m|s - \bar{s} - f^{-1}(s) + f^{-1}(s_s)| \leq m|s - \bar{s}|, \qquad s, \bar{s} \geq 0.$$

Using the Contraction Mapping Theorem [17, p. 126] we conclude that the s_k converges to the unique fixed point s_∞ of (10), which is the unique solution to

$$s_\infty = m\big(s_\infty - f^{-1}(s_\infty)\big) + \frac{1-m}{an}(q_{\max} + BT_p) - m. \tag{11}$$

The convergence is as fast as m^k. From this and (9) we conclude that the following theorem holds:

Theorem 1. *For $q_{\max} + BT_p \geq \frac{2ma}{1-m}n$, all the $w_i(\tau_k)$, $i \in \{1, 2, \ldots, n\}$, converge exponentially fast to*

$$w_\infty := \frac{ma}{1-m}(f^{-1}(s_\infty) + 1), \tag{12}$$

as $k \to \infty$. The convergence is as fast as m^k.

A straightforward conclusion of Theorem 1 is that all the flows become synchronized as time goes to infinity. This is because the window sizes of all the flows converge to the same limit cycle. This limit cycle correspond to an increase of the window size from w_∞ to $\frac{1}{m}w_\infty$, lasting $f^{-1}(s_\infty) + 1$ units of normalized time, followed by an instantaneous decrease back to w_∞ due to drops.

4 Steady-state behavior

We proceed now to derive a formula, such as the ones derived in [2, 3], that relates the average throughput with the average *drop rate* (i.e., the percentage of dropped packets) and the average round-trip time. To this effect, suppose that the steady state has been reached and therefore

$$\tau_{k+1} - \tau_k = f^{-1}(s_\infty) + 1, \tag{13}$$

$$t_{k+1} - t_k = \frac{man}{B(1-m)}f^{-1}(s_\infty)(f^{-1}(s_\infty) + 1) + \frac{an}{2B}f^{-1}(s_\infty)^2 + \frac{man}{B(1-m)}\big(f^{-1}(s_\infty) + 1\big) + \frac{ans_\infty}{B}. \tag{14}$$

Here, we used (8), (12) and the fact that

$$q_{\max} + BT_p = \frac{man}{1-m}\big(f^{-1}(s_\infty) + 1\big) + ans_\infty,$$

which can be derived from (11).

We concentrate next on the case where $f^{-1}(s_\infty)$ is much larger than one. This occurs when the system remains in the state *queue-not-full* for a period of time considerably larger than one round-trip time. For $s_\infty \gg 1$, it is straightforward to show hat $f^{-1}(s_\infty) \approx s_\infty + 1 \approx s_\infty$ and therefore

$$\tau_{k+1} - \tau_k \approx s_\infty, \qquad t_{k+1} - t_k \approx \frac{an}{2B}\frac{1+m}{1-m}s_\infty^2. \tag{15}$$

In each interval $[t_k, t_{k+1})$ there are n drops out of $B(t_{k+1} - t_k)$ packets sent. Therefore the average drop rate p is equal to

$$p := \frac{n}{B(t_{k+1} - t_k)} \approx \frac{2}{a} \frac{1-m}{1+m} \frac{1}{s_\infty^2}. \tag{16}$$

On the other hand, since each unit of normalized time corresponds to one round-trip, the average round-trip-time is equal to

$$\overline{RTT} := \frac{t_{k+1} - t_k}{\tau_{k+1} - \tau_k} \approx \frac{an}{2B} \frac{1+m}{1-m} s_\infty. \tag{17}$$

We therefore conclude that

$$\frac{1}{\overline{RTT}\sqrt{p}} = \sqrt{\frac{B(\tau_{k+1} - \tau_k)^2}{n(t_{k+1} - t_k)}} \approx \sqrt{\frac{2}{a} \frac{1-m}{1+m}} T,$$

where $T := \frac{B}{n}$ is the average throughput of each flow, which means that

$$T \approx \sqrt{\frac{a}{2} \frac{1+m}{1-m}} \frac{1}{\overline{RTT}\sqrt{p}}. \tag{18}$$

For $a = 1$ and $m = 1/2$ we obtain

$$T \approx \frac{\sqrt{3/2}}{\overline{RTT}\sqrt{p}},$$

which confirms similar formulas derived in [2, 3]. It should be noted that the derivations in these references do not take queueing into account. The coupling between the n competing flows is also ignored and therefore no theoretically-supported claim is made to the extent that the steady state solution is actually reached in an asymptotic sense. Moreover, in these references no formulas are derived for $\overline{RTT}$ and p as a function of the number of flows n, such as the ones in (16) and (17).

Formulas for $\overline{RTT}$ and p can also be derived directly from (13) and (14) for cases when s_∞ is not much larger than one. However, this situation usually corresponds to unusually high drops rates that only occur for very large number of flows.

To test the formulas derived above we ran several simulations of a network with the dumbbell topology using the network simulator ns-2 [18]. Figure 4 summarizes the results obtained for a network with the following parameters: $B = \frac{10^7 \text{ bits/sec}}{8 \text{ bits/char} \times 1000 \text{ char/packet}} = 1250$ bits/packet, $T_p = .04$ sec, $q_{\max} = 250$ packtes, $a = 1$ packet/RTT, $m = 1/2$. As seen in the figure, the theoretical predictions given by (16)–(18) match the simulation results quite accurately. Some mismatch can be observed for large number of flows. However, this mismatch only starts to become significant when the drop rates are around 1%, which is an unusually large value. This mismatch is due to the existence of multiple drops per flow (cf. Remark 1).

5 Conclusions

In this paper we proposed a hybrid model for Reno congestion control. Using this model, we analyzed both the transient and the steady state behavior of n TCP flows competing for the available bandwidth on a dumbbell network topology. Our model confirmed formulas for the steady-state behavior that could be found in the literature and also derive new relationships between the several quantities of interest. We were also able to explain the flow synchronization phenomena that has been observed in simulations and in real networks but, to the best of our knowledge, has not been theoretically justified. In are now in the process of exploring mechanisms that can be used to avoid the undesirable synchronization. Another application of the hybrid model derived here is the detection of abnormalities in TCP traffic flows. This has important applications in network security.

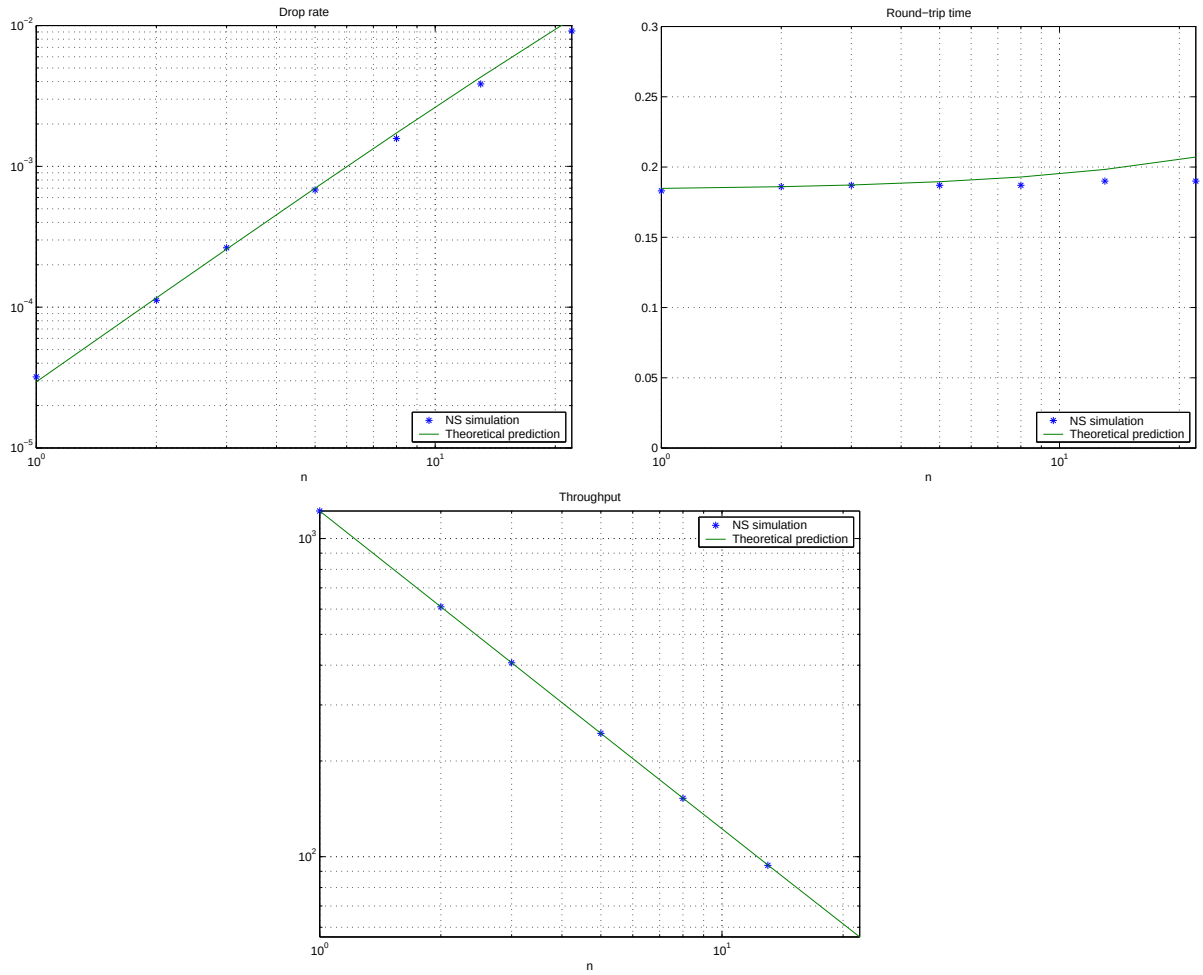

Figure 4: Comparison between the predictions obtained from the hybrid model and the results from ns-2 simulations.

References

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

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

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

[4] S. Floyd, M. Handley, J. Padhey, and J. Widmer, "Equation-based congestion control for unicast applications." To appear in SIGCOMM, May 2000.

[5] S. Floyd and V. Jacobson, "On traffic phase effects in packet-switched gateways," *Internetworking: Research and Experience*, vol. 3, pp. 115–116, Sept. 1992.

[6] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. on Networking*, vol. 1, pp. 397–413, Aug. 1993.

[7] L. Tavernini, "Differential automata and their discrete simulators," *Nonlinear Anal. Theory, Methods, and Applications*, vol. 11, no. 6, pp. 665–683, 1987.

[8] A. S. Morse, D. Q. Mayne, and G. C. Goodwin, "Applications of hysteresis switching in parameter adaptive control," *IEEE Trans. Automat. Contr.*, vol. 37, pp. 1343–1354, Sept. 1992.

[9] A. Back, J. Guckenheimer, and M. Myers, "A dynamical simulation facility for hybrid systems," in Grossman *et al.* [19].

[10] A. Nerode and W. Kohn, "Models for hybrid systems: Automata, topologies, stability," in Grossman *et al.* [19], pp. 317–356.

[11] P. J. Antsaklis, J. A. Stiver, and M. D. Lemmon, "Hybrid system modeling and autonomous control systems," in Grossman *et al.* [19], pp. 366–392.

[12] R. W. Brockett, "Hybrid models for motion control systems," in *Essays in Control: Perspectives in the Theory and its Applications* (H. L. Trentelman and J. C. Willems, eds.), pp. 29–53, Boston: Birkhäuser, 1993.

[13] M. S. Branicky, V. S. Borkar, and S. K. Mitter, "A unified framework for hybrid control: Background, model and theory," in *Proc. of the 33rd Conf. on Decision and Contr.*, vol. 4, pp. 4228–4234, Dec. 1994.

[14] M. S. Branicky, *Studies in Hybrid Systems: Modeling, Analysis, and Control.* PhD thesis, MIT, Cambridge, MA, June 1995.

[15] J. Lygeros, C. Tomlin, and S. Sastry, "Multi-objective hybrid controller synthesis: Least restrictive control," in *Proc. of the 36th Conf. on Decision and Contr.*, vol. 1, pp. 127–132, Dec. 1997.

[16] J. P. Hespanha, S. Bohacek, K. Obraczka, and J. Lee, "Hybrid modeling of tcp congestion control," tech. rep., University of Southern California, Los Angeles, CA, Oct. 2000.

[17] 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.

[18] 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.

[19] R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rishel, eds., *Hybrid Systems*, vol. 736 of *Lecture Notes in Computer Science.* New York: Springer-Verlag, 1993.