Modeling the Effect of Short-term Rate Variations on TCP-Friendly Congestion Control Behavior

Kang Li[℘], Molly H. Shor[±], Jonathan Walpole[℘], Calton Pu[ℵ], David C. Steere[℘]

Abstract:

Transmission Control Protocol (TCP) is the dominant transport protocol in today's Internet. To maintain the stability of the Internet, flows other than TCP need to be "friendly" to TCP flows. The "TCP-friendliness" of a protocol for the transmission of data over a network refers to whether it shares network bandwidth fairly with TCP traffic. The criterion generally used for TCP-friendliness is "fair share" of long-term average throughput.

While existing network simulators for TCP and TCP-friendly control work well in general, they can not make theoretical proofs of the stability of the bandwidth-sharing behavior by competing TCP-friendly congestion controls. To study the effect on bandwidth sharing of interactions among a set of competing TCP-friendly flows, we built a hybrid state-space-based model of TCP, using differential equations and event-driven switches. Our research focuses on modeling a class of TCP-friendly congestion controls that use the additive-increase and multiplicative-decrease (AIMD) algorithm for congestion avoidance but also use a rate smoother. The rate smoother can produce various short-term transmission rates that are different from normal TCP.

With our model, we prove that TCP-friendly flows result in a stable attractor if the backing off of flow transmission rates is synchronized. Experiments using our model with unsynchronized backing off show unfairness among competing flows. This result indicates that, beyond TCP-friendliness, we need other metrics to measure a flow to determine whether it can share bandwidth evenly with other competing TCP traffic.

1. Introduction

One of the most common "protocols" used to control the transmission of data in a reliable fashion on the Internet is the Transmission Control Protocol (TCP) [1]. This protocol adjusts how much data may be sent for a particular connection between end applications over each interval of time. TCP is designed to "back off" the rate of transmission when network congestion occurs and to increase the rate over time when there is no congestion. The increase is TCP's mechanism to probe the network to determine if excess capacity is available.

TCP congestion control can be viewed as a nonlinear feedback control system that dynamically adjusts its transmission rate according to the network's congestion state. A significant amount of research work has been done on this system. For example, Jain and Chiu [2] proved that multiple TCPs converge to fair bandwidth share by analyzing TCP's additive-increase and multiplicative-decrease (AIMD) algorithm. Padhye, et al., [3, 4, 5] derived equations for TCP's average transmission rate from the system assuming that the TCP's feedback delay (round-trip-time) and packet loss rate are known.

Recently, new "TCP-friendly" protocols were proposed to serve the emerging class of multimedia applications. While TCP is appropriate for applications such as bulk data transfer, it does not serve well streaming multimedia applications. TCP's variable delay, caused by retransmission and rate variations, may reduce the user's perceived quality [6]. Various TCP-friendly congestion controls have been proposed as alternatives to TCP. A flow is TCP-friendly if its long-term average transmission rate is equal to that of a normal TCP under the same conditions (e.g., round-trip-time (RTT) and packet loss rate).

[℘] Dept. of Computer Science and Engineering, Oregon Graduate Institute, Beaverton, Oregon, USA. Research was supported in part by DARPA Grant N66001-00-2-8901, NSF Grant CCR-9988440.

[±] Dept. of Electrical and Computer Engineering, Oregon State University, Corvallis, Oregon, USA. Research was supported in part by NSF Grant ECS-9988435.

[ℵ] College of Computing, Georgia Institute of Technology, Atlanta, Georgia, USA

While traditional network simulators [7] work well in general, it is not possible to use them for theoretical proofs of the stability of the bandwidth-sharing behavior by competing TCP-friendly congestion controls. In this paper, we use a hybrid state-space-based model to investigate the bandwidth-sharing behavior of TCP-friendly flows. We choose the transmission rates of TCP-friendly flows, as well as the network buffer fill level, as system states to describe the bandwidth sharing behavior, and we prove that the system state converges to a stable limit cycle under the assumption of synchronous backing off of all flows.

We also compare TCP-friendly flow behaviors under other assumptions on congestion signals. Our experiment with asynchronous backing off indicates that the definition of TCP-friendliness (in terms of long-term average throughput) does not guarantee fair bandwidth sharing among competing TCP-friendly flows.

The remainder of the paper is organized as follows. Section 2 provides an overview of the TCP congestion control system. Section 3 presents the current definition and models for TCP-friendliness. Section 4 describes our state-space model and our theoretical analysis of the stability of bandwidth sharing among TCP-friendly flows. We present the experiments through simulations at the end of Section 4 and some future work in Section 5, and Section 6 concludes the paper.

2. TCP AIMD Control as a Nonlinear Feedback-based Control System

TCP is an adaptive protocol, using the additive-increase multiplicative-decrease algorithm. TCP adjusts the sender's data output rate according to the network path characteristics and the receiver side behaviors.

We can represent TCP congestion control as a feedback control system that outputs a signal onto the network to probe the network state, which is then used to control the data output rate. This feedback control system is illustrated in Figure 2.1. The feedback loop is composed of the rate controller, the probing signal that goes across the network, and the feedback monitor that monitors the sampling results and sends them to the rate controller.

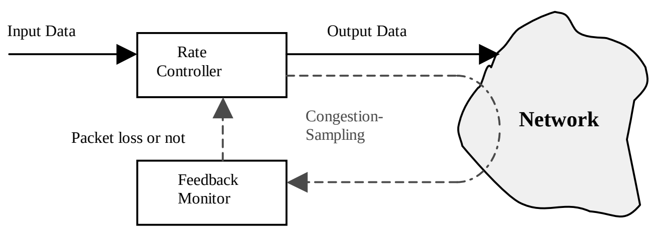

Figure 2.1: TCP Congestion Control System

TCP probes the network's state with the data it sends. Data packets travel from the sender to the receiver, and acknowledgments for each packet travel back from the receiver to the sender.

The time from sending a packet to receiving its acknowledgment is the round-trip time (RTT). The RTT is an important state variable in this system. This is the delay around the feedback loop, as well. The RTT varies primarily as a function of the buffer fill levels in the network path along which the data travels. The longer the packets must wait in buffers, the longer it takes them to traverse that path. A significant increase in RTT may be a useful indicator of network congestion.

If a packet arrives somewhere and the buffer is full, then it is lost. Each time the rate controller probes the network, the feedback monitor determines if the packet's acknowledgment returns or not. TCP detects this packet loss by looking at out-of-order acknowledgments. If acknowledgments have arrived for three packets that are sent out later than a certain packet that has not been acknowledged, then TCP decides that that packet is lost.

TCP controls the rate at which data is sent out on the network by using a congestion window. The congestion window size defines the maximum amount of outstanding data, data that has been sent but not yet acknowledged; hence, the amount that is sent out in one round-trip-time. The congestion window size is an important state variable in this system. The rate controller uses an AIMD algorithm to control the congestion window size, or out-going data rate. If acknowledgments are received for all packets that are sent during one RTT, then TCP increases its congestion window size by α (default is one packet); otherwise TCP decreases its congestion window to β times (default is half) the current window size.

3. TCP-friendliness

TCP-friendliness is proposed because TCP is not well suited for emerging applications such as streaming media. TCP integrates congestion control and reliability control together. The reliability control usually causes too much latency for applications that are very sensitive to delay and delay variations but can tolerate some data loss. When transferring streaming media with TCP, TCP's saw-tooth rate variation may also cause undesired delay and delay variations. TCP-friendly congestion controls are suited for these timing sensitive applications and interact well with TCP. They ensure that normal TCP flows get their fair allocation of bandwidth in the presence of these protocols, and vice versa.

Right now, TCP-friendliness [3, 8, 9, and 10] is defined based on TCP's long-term average throughput behavior. A well-known behavior of TCP is that the long-term (at least several seconds) average throughput of a flow with TCP AIMD congestion control is related to its loss rate, round-trip-time, and maximum packet size [8, 11, and 5] according to the formula:

$$\bar{r} = \frac{1.22M}{RTT * \sqrt{p}} \qquad \text{Equation (3.1).}$$

The terms used in the Equation (3.1) are defined in Table 3.1.

$\bar{r}$: A TCP's average throughput;
p: A TCP's packet loss rate;
r(t): A TCP's transmission rate at time t;
fl(t): Network leaky bucket queue fill-level at time t;
α : TCP AIMD's linear increment parameter (default 1), and $\alpha \geq 0$;
β : TCP AIMD's exponential decrement parameter (default ½), and $0 < \beta < 1$;
M: TCP's maximum segment size (We assume all TCP packets are this size);
RTT: TCP's round-trip-time (We assume RTT is a constant for a flow);
R: Network leaky bucket's leaking rate;
B: Network leaky bucket's bucket size;
N: Number of competing flows.

Table-3.1: Term Definitions

TCP-friendliness is generally defined by the average throughput specified in Equation (3.1). If a non-TCP flow's long-term average throughput is equal to the throughput specified by Equation (3.1), then it is TCP-friendly. The short-term behavior of a TCP-friendly flow is left unspecified, because one motivation for proposing TCP-friendly protocols is to provide alternative short-term transmission rates that differ from TCP's saw-tooth rate variation.

According to the above definition of TCP-friendliness, various mechanisms have been proposed to implement TCP-friendly congestion controls. One way to implement a TCP-friendly congestion control with a different short-term rate behavior is to use TCP's AIMD algorithm with different increment α and decrement parameters β with the parameters related by equations such as

$$\alpha = \frac{3(1-\beta)}{1+\beta} \qquad \text{Equation (3.2)}$$

Another way to implement a TCP-friendly congestion control is to monitor the packet loss rate and round-trip-time and calculate the average throughput according to Equation (3.1). Once the desired average rate is known, the congestion control can use the inverse of the rate to control the interval between outgoing packets.

The TCP-friendliness definition based on the long-term average throughput of a flow operating with a given loss rate makes a potential assumption that a flow's packet loss rate is independent on its short-term rate variations. If loss event (congestion signal) distributions are the same for all the competing flows, they would certainly get the same bandwidth share according to the equation. However, the reality may differ from this assumption. Existing TCP-friendly congestion control protocols may not use the throughput equation (e.g., TCP using different AIMD parameters), or they may use the throughput equation but have different ways to measure the packet loss rate. Experiments [10] have shown that various "TCP-friendly" flows actually will not share bandwidth equally with each other even when the network set up (same RTT and same bottleneck link) for those flows are the same, although they are supposed to share bandwidth evenly in long-term according to the friendliness definition. Therefore, we think it worthwhile to study the behavior of bandwidth sharing among multiple competing TCP flows.

4. State Space Based Model of TCP

This section discusses a state-space based model for TCP, and presents a result derived using the model.

We chose to derive a state-space model to study the bandwidth-sharing behavior under the effect of short-term rate variation. The reason we chose this model is that we believe that the congestion signal distribution is related to the instantaneous transmission rates when congestion occurs. Therefore, we need to have a model that includes every competing flow's instantaneous transmission rate and the queue fill-level of the bottleneck link[1].

Our state-space based model includes all the important states that determine the bandwidth-sharing behavior. The essential feature of the concept of state for a dynamical system is that it should contain all information about the past history of the system that is relevant to its future behavior. That is to say, if the state at a given instance is known, then its subsequent evolution can be predicted without any other knowledge of what has previously happened to the system. For a system with *N* competing flows, the system's state at time t in our model is a vector $S(t) = [r_1(t), r_2(t), \cdots, r_N(t), fl(t)]^T$, where the terms are defined in Table 3.1.

For each individual flow, we chose the following state equations to describe the dynamic states of a TCP-friendly flow that uses the AIMD algorithm. We assume TCP has a constant RTT, and we use a continuous model for the increase to approximate the rate increment by one packet per RTT. Since the AIMD algorithm switches its rate control according to the existence of a congestion signal, we use two equations to describe the system behavior:

When no congestion happens (in uncongested state):

$$\frac{dr(t)}{dt} = \frac{\alpha M}{RTT^2} \qquad \text{Equation (4.1),}$$

When congestion happens:

$$r \leftarrow \beta \times r \qquad \text{Equation (4.2).}$$

We simulate a bottleneck link with a leaky bucket model. The bucket fill-level is controlled by

$$\frac{dfl(t)}{dt} = R - r \qquad \text{Equation (4.3).}$$

When *fl(t) = B*, a congestion signal is produced.

4.1 Trajectory, limit cycle, and attractor

We start this section by introducing some terms that are widely used in describing nonlinear control systems. The state space defined by system states is called the *phase plane*. A system's *trajectory* describes how the system states migrate in the phase plane. If the system states can fully capture the system's dynamics (e.g., in a deterministic, homogenous, time-invariant system), then any point in the phase plane determines the system's future behavior, and the next state. Therefore trajectories cannot cross. Some special trajectories take the form of closed loops, which we call *limit cycle*s. A limit cycle is called stable, or an *attractor*, when its nearby trajectories converge to the limit cycle. Figure 4.1 provides an example of a trajectory of the system of interest with two competing flows. The system state starts at a random point in the state space, and the trajectory converges to a stable limit cycle (attractor).

---

[1] We assume only one, common, bottleneck for all the competing flows.

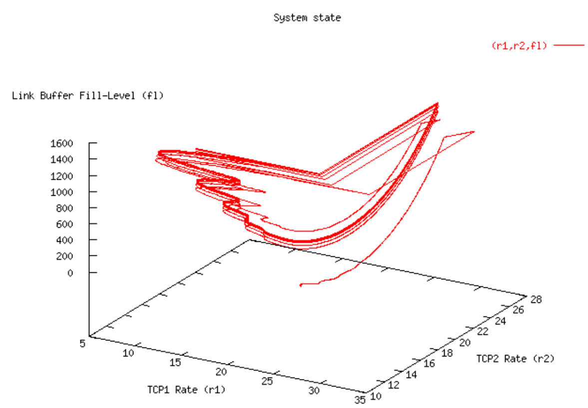

Figure 4.1: The trajectory of a system that has two competing flows.

4.2 The attractor of competing TCP-friendly flows under synchronized backing off

Theorem 1:

When multiple TCP-friendly flows compete for a constant available bandwidth, the system states converge to a limit cycle that passes through the point $P = [r_1, r_2, \cdots, r_N, B]^T$ *, in which*

$$r_i = \frac{2\beta}{1+\beta} \times \frac{1}{RTT_i^2} \times \frac{R}{\sum_{j=1}^{N} \frac{1}{RTT_j^2}} \qquad \textit{Equation (4.4).}$$

The long-term average throughput of a flow is

$$\bar{r}_i = \frac{1}{RTT_i^2} \times \frac{R}{\sum_{j=1}^{N} \frac{1}{RTT_j^2}} \qquad \textit{Equation (4.5).}$$

When all the flows have the same RTT, they get even bandwidth share.

Here, we argue that the above limit cycle is stable. The full proof for the theorem is presented in a complete version of this paper [12].

To show that the limit cycle is stable, we divide the system trajectory into rounds. Each round starts with a backing off upon a congestion signal, and ends with producing the next congestion signal. We compare the distance of the system state from the above limit cycle at the beginning of every two consecutive rounds. If the distance is converging to zero, it indicates that the system state vector is approaching the limit cycle. Furthermore, we know if the system state ever reaches the above limit cycle, it will stay on it, unless there is noise to cause the system state to leave the limit cycle again.

We assume that the system states of the flows start from a random initial condition $P' = [r_1 + \Delta'_1, r_2 + \Delta'_2, \cdots, r_N + \Delta'_N, B]^T$, and the system states arrive at $P'' = [r_1 + \Delta''_1, r_2 + \Delta''_2, \cdots, r_N + \Delta''_N, B]^T$ after one more round. We can derive the following relation:

$$\Delta''_i = \beta \Delta'_i - \frac{1}{RTT_i^2} \times \frac{2\beta}{\sum_{j=1}^{N} \frac{1}{RTT_j^2}} \sum_{j=1}^{N} \Delta'_j \qquad \text{Equation (4.6)}$$

We can prove that $\frac{\sum_{i=1}^{N}(\Delta''_j)^2}{\sum_{i=1}^{N}(\Delta'_j)^2} < 1$ when $0 < \beta < 1$, which indicates that the distance to the limit cycle becomes smaller as time goes by. Therefore, we know that the product of that formula with the next formula, etc., will result in the product of various numbers that are all less than one. As time goes to infinity, the distance to the limit cycle approaches zero. Thus, the limit cycle is stable.

We have presented elsewhere that the trajectory of a system composed by a single TCP is a limit cycle [13]. Here we have proved that a system with multiple flows converges to a stable limit cycle when the flows' average throughputs are equal to a normal TCP's throughput, assuming all the flows back off together. This indicates that the TCP-friendliness definition is enough to guarantee fair bandwidth share among competing traffic under a synchronized feedback.

4.3 Asynchronous backing off

In this section, we discuss the bandwidth sharing behavior of competing TCP-friendly flows under asynchronous backing off.

We use a rate-smoother to produce TCP-friendly flows with various short-term transmission rates. The rate smoother is built on a rate-matching control system. Figure 4.2 shows the structure of a rate-matching control system. The mismatch window records the accumulated mismatch of its input and output rate. The PID control can be described by the equation:

$$u(t) = Pe(t) + I\int_0^t e(\eta)d\eta + D\dot{e}(t) \qquad \text{Equation (4.7),}$$

in which e(t) is the accumulated rate mismatch; u(t) is the output rate; and P, I, and D are control parameters of the controller. To produce various TCP-friendly flows, we let the input to the mismatch window be the rate produced by the TCP AIMD control or by the throughput equation (3.1), and use the output rate as the flow's transmission rate. By adjusting the PID control parameters, we can produce various short-term rate variations but maintain the same long-term average throughput as TCP.

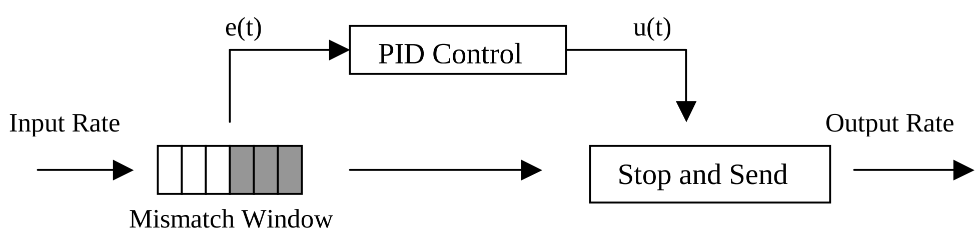

Figure 4.2 A Rate-Matching Control System

Real network measurement [14] indicates that packet loss events are often not evenly distributed to all competing flows. In reality, flows with high transmission rates are more likely to get packet loss events when congestion happens. Furthermore, the loss event distribution is not necessarily proportional to the transmission rate [8].

To investigate the possible effect of short-term rate variation on the bandwidth sharing behavior, we chose a set of loss event distributions to study the bandwidth-sharing ratio among competing flows. Since we believe the loss event distribution can vary a lot on different paths, we intended to choose a set of loss event distributions that cover a wide range from uniform distribution to proportional distribution. In our experiments, the set of loss event distributions is controlled by the function of $p = 1 - a * Exp(-b * r)$. The relation between the distribution and parameters (*a*, *b*) are shown in Figure 4.3 (a).

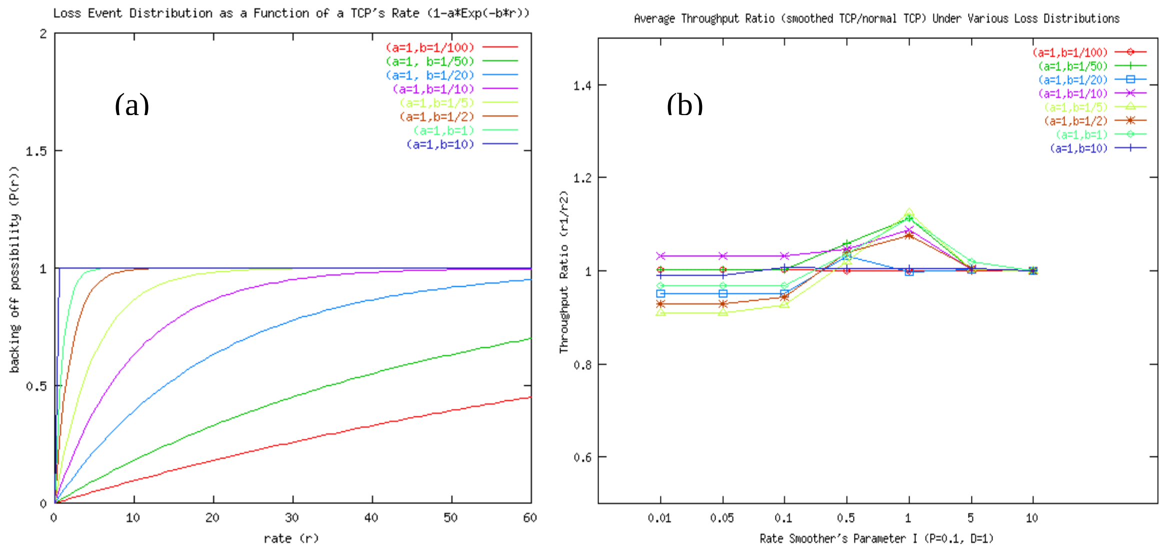

Figure 4.3 Loss Event Distributions and Bandwidth Sharing Ratio

Figure 4.3 shows our simulation results with various loss event distributions. The simulations are made by implementing our model using Matlab/Simulink [15], and the simulation files are available through the web [16]. Figure 4.3 (b) shows the bandwidth share ratio by two competing flows. One flow is a normal AIMD TCP; another one is a TCP-friendly flow that is produced by an AIMD TCP but with a rate smoother that changes its short-term rate behavior. We adjust the rate smoother's parameters to adjust the short-term behavior. As a result, we have varying bandwidth sharing ratios.

Clearly, from our simulation results, short-term rate variation has an effect on the bandwidth sharing results as well as the loss event distributions.

When the rate smoother's output tracks the original TCP's rate closer (when I>5 and P=0.1, D=1), the two flows are closer to even bandwidth share. This is because the two flows will not have much different transmission rates. When the two flows' short-term behaviors diverge, the bandwidth share is diverges from even share.

When the loss event distribution is close to a uniform distribution, the two flows are close to having synchronized backing off. Therefore, their bandwidth shares do not have much difference. When the loss event distribution is proportional to the transmission rate, the bandwidth share ratio is also close to 1. When the distribution is in between, short-term bursts make the two flows share bandwidth differently. We plan to quantify the effect of bursts on bandwidth share ratio in the future.

5. Factors beyond TCP-friendliness

Clearly, the current interpretation of TCP friendliness is not adequate to determine which congestion controls share network bandwidth fairly with one another and TCP. That raises the question: what factors beyond TCP-friendliness have significant impact on bandwidth sharing behavior?

Differing loss event distributions causes a congestion-controlled flow to consume a different bandwidth share than another flow. For the loss process of an ordinary FCFS queue, the only difference between two flows is their rates. Floyd, et al., [9] proposed the notion of TCP-friendliness, which is the first order rule to keep fair bandwidth share between new flows and traditional TCP traffic, but captures only the average of a flow's transmission rate. We believe that other aspects of the transmission rate, such as burst and oscillation frequency, could be important factors that can be used to predict or control the bandwidth-sharing behavior.

Burst can be measured by the maximum accumulated mismatch of a flow's real transmission rate and the "TCP-friendly" average throughput. A flow can burst out some data and keep the same average throughput by borrowing some amount from the average and returning it later (by sending less than average). Limits on burst are limits on the maximum amount that a flow can borrow, which we believe is one factor that affects flow transmission rates.

Oscillation frequency describes how frequently a flow varies (borrows and returns) from its average throughput. We believe this is another factor that affects flow transmission rates.

We have not quantified the effect on bandwidth share caused by either of the above factors. We will continue to study these short-term rate variations on the bandwidth sharing result. Other future work includes how to control these parameters to obtain a user's preferred result (such as proportional bandwidth share rather than equal share).

6. Conclusion

In this paper, we focused on studying the effect of short-term rate variations on bandwidth sharing behavior. Currently, TCP-friendliness is the well-accepted notion used to build new congestion-control protocols that keep bandwidth fair share. At the same time, several research works have shown that the notion itself is not enough to guarantee bandwidth fair share. In our work, we built a state-space-based model and proved the stability of bandwidth sharing behavior under the assumption of synchronized backing off.

We also seek to understand the unfair share of bandwidth under asynchronous backing off and to propose some possible factors to study in the future. As we have seen, understanding bandwidth sharing among competing traffic is complicated. We believe that the problem of predicting and controlling the bandwidth sharing among competing flows is an important task for both the current and future Internet as long as the network resources are shared among users. To fully understand the problem requires more research work.

Reference

[1] Van Jacobson, and Michael J. Karels. "Congestion Avoidance and Control". Proceeding of ACM SIG-COMM'88, pp. 79-88, August 1988.

[2] D. Chiu and R. Jain. "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks".

[3] Jitendra Padhye, Jim Kurose, Don Towsley, and Rajeev Koodli. "A Model Based TCP-Friendly Rate Control Protocol". UMass-CMPSCI Technical Report TR 98-04, October 1998.

[4] Jitendra Padhye, Victor Firoiu, and Don Towsley. "A Stochastic Model of TCP Reno Congestion Avoidance and Control". CMPSCI Technical Report 99-02.

[5] Jitendra Padhye, Victor Firoiu, Don Towsley and Jim Kurose. "Modeling TCP Throughput: A Simple Model and its Empirical Validation". ACM SICOMM'98.

[6] S. Cen, C. Pu and J. Walpole. "Flow and Congestion Control for Internet Streaming Applications", in the proceedings Multimedia Computing and Networking 1998 (MMCN98).

[7] NS2 simulator. Available at http://www-mash.CS.Berkeley.EDU/ns/

[8] S. Floyd, M. Handley, and J. Padhye. "Equation-Based Congestion Control for Unicast Applications". SIGCOMM'2000.

[9] Sally Floyd, and Kevin Fall. "Promoting the Use of End-to-End Congestion Control in the Internet" IEEE/ACM Transactions on Networking, August 1999. Available at http://www.aciri.org/floyd/papers.html

[10] S. Floyd, M. Handley, and J. Padhye. "A comparison of Equation-based and AIMD Congestion Control", May 2000. Available on http://www.aciri.org/tfrc/

[11] Matthew Mathis, Jeffrey Semke, and Jamshid Mahdavi. "The Macrosopic Behavior of the TCP Congestion Avoidance Algorithm". ACM Computer Communication Review, vol 27 num 3, July 1997. Available at http://www.psc.edu/networking/papers/model_abstract.html

[12] K. Li, M. Shor, and J. Walpole. “OGI Tech-Report: Modeling the Effect of Short-term Rate Variations on TCP-Friendly Congestion Behavior”. Available at http://www.cse.ogi.edu/~kangli/

[13] M. Shor, K. Li, J. Walpole, D. C. Steere, and C. Pu. "Application of Control Theory to Modeling and Analysis of Computer System". June 7-9, 2000.

[14] V. Paxson. “End-to-End Internet Packet Dynamics.” IEEE/ACM Transactions on Networking, 7(3), pp. 277-292, June 1999.

[15] Matlab Simulink. The MathWorks Inc. http://www.mathworks.com/products/simulink/

[16] Simulation files for the State-Space model. http://www.cse.ogi.edu/~kangli/simulink.html