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

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

Abstract:
Transmission Control Protocol (TCP) is the dominant transport protocol in today's Internet. To maintain stability of the Internet, flows other than TCP must be "friendly" to TCP flows, or share network bandwidth fairly with TCP traffic. Usually a flow is claimed to be TCP-friendly when its throughput is theoretically the same as the throughput of a TCP flow when they experience the same congestion signals. However, when flows compete for bandwidth, they may not have the same perception of congestion. Therefore, measured bandwidth shares of flows are not necessarily equal, even when all flows are theoretically designed to be TCP-friendly.

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. We modified the TCP model, using TCP's additive-increase multiplicative-decrease (AIMD) congestion avoidance algorithm with different increase and decrease parameters, to create theoretically TCP-friendly protocols with various short-term transmission rates. 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 and using ns simulator with unsynchronized backing off show unfairness among competing flows with different short-term behaviors.

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 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 support the requirements of streaming multimedia applications. TCP's variable delay, caused by retransmission and rate variations, may reduce a user's perceived quality [6]. Various TCP-friendly congestion controls have been proposed as alternatives to TCP. A flow is generally claimed to be 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).

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 then compare TCP-friendly flow behaviors under other assumptions on congestion signals. We made simulation experiments with our model, as well as with ns simulator [7]. Our experiments with asynchronous backing off indicate that theoretically TCP-friendly flows do not share bandwidth fairly in practice, because they do not perceive congestion in the same way.

This result confirms that a theoretical TCP-friendly throughput prediction does not guarantee even bandwidth share among competing flows in reality, and indicates that

[℘] Dept. of Computer Science and Engineering, Oregon Graduate Institute, Beaverton, Oregon 97921-1000 USA. This work was supported in part by DARPA/ITO under the Information Technology Expeditions, Ubiquitous Computing, Quorum, and PCES programs, and in part by NSF Grant CCR-9988440.
[Ψ] Dept. of Electrical and Computer Engineering, Oregon State University, Corvallis, Oregon 97331-3211 USA. Research was supported in part by NSF Grant ECS-9988435
[⊗] College of Computing, CCB Room 261, Georgia Institute of Technology, Atlanta, Georgia 30332-0280 USA

we need additional metrics beyond theoretical throughput to determine whether a flow will share bandwidth evenly with competing TCP traffic.

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, our theoretical analysis of the stability of bandwidth sharing among TCP-friendly flows, and our experiments through simulations. Sections 5 and 6 discuss future work and conclusions.

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.

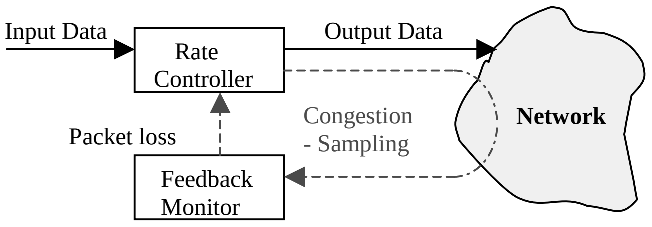

Figure 2.1: TCP Congestion Control System

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:

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

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

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 called 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.

Table 3.1: Term Definitions

$\bar{r}$: 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.

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 [17]

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

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 the assumption that a flow's packet loss rate is independent on its short-term rate variations. If loss event (congestion signal) distributions were the same for all the competing flows, they would certainly get the same bandwidth share according to the equation. However, reality may differ from this assumption. Some existing TCP-friendly congestion control protocols do not use the throughput equation (e.g., TCP using different AIMD parameters), and some 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 for those flows are the same (same RTT and same bottleneck link), 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 presents a state-space-based model for TCP, a result derived using the model, and simulation results.

We chose to derive a state-space model to study the bandwidth-sharing behavior under the effect of short-term rate variation. We believe that the congestion signal distribution is related to the instantaneous transmission rates when congestion occurs. Therefore, we required 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[2], 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 the following equations (transitions) to describe the system behavior:

When no congestion occurs (uncongested state):

$$\frac{dr_i(t)}{dt} = \frac{\alpha M}{RTT_i^2} \qquad (4.1),$$

When congestion occurs:

$$r_i \leftarrow \beta \times r_i \qquad (4.2).$$

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

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

[2] RTT is composed of propagation delay, which is a constant, and queueing delay, which is related to queue fill-level. Here we assume the propagation delay is much larger than queueing delay to simplify the description. An extended model with RTT as a variable is in [12].

$$\begin{cases} \dfrac{dfl(t)}{dt} = \sum_{i=1}^{N} r_i - R & \text{if } fl(t) > 0 \\ \dfrac{dfl(t)}{dt} = \max(\sum_{i=1}^{N} r_i - R, 0) & \text{if } fl(t) = 0 \end{cases} \quad (4.3).$$

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

Since *fl(t)* is the fill-level of the bottleneck, we expect *fl(t)* > 0 always and hence neglect the *fl(t)* = 0 case in Equation (4.3) in our analysis.

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).

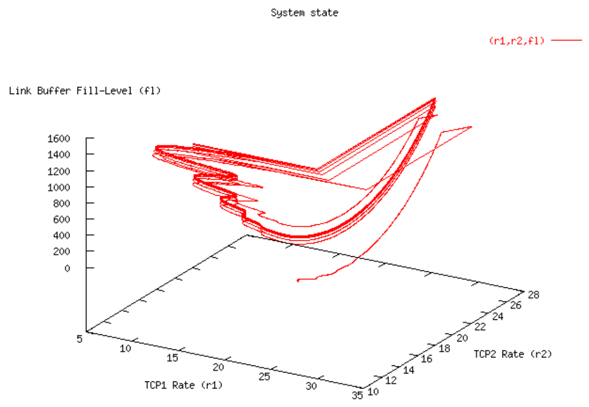

Figure 4.1: The trajectory of a system that has two competing flows.

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

Next, we present a theorem on the stability of the limit cycle that results 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*

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

The long-term average throughput of a flow is

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

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:

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

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 adjust TCP's AIMD parameters according to Equation (3.2) to create a TCP-friendly flow with different short-term rate behavior but the same long-term throughput in theory [17]. A larger α causes a more aggressive short-term rate increase. Due to the theoretical limitation of keeping the same throughput as normal TCP, β must also be larger, which causes a greater short-term rate decrement when congestion occurs.

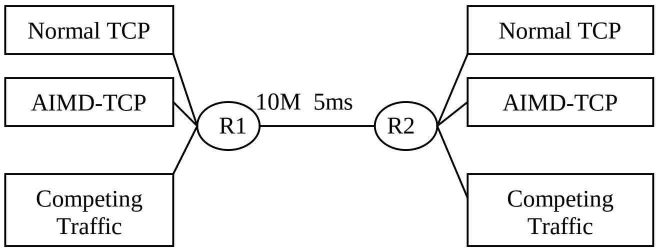

Figure 4.2: Simulation Topology

We study the impact of these short-term rate behaviors under asynchronous backing off using simulations of our model, implementing it in Matlab/Simulink [15]. The simulation files are available through the web [16].

The simulations use the network setup shown in Figure 4.2. In this setup, two TCP flows share a bottleneck link with some competing traffic. One TCP flow uses the standard AIMD parameters (1,1/2). The other TCP uses the modified AIMD parameters. We conducted the experiment with the following pairs of AIMD parameters (α, β): (1/3, 4/5), (2/3, 7/11), (4/3, 5/13), and (5/3, 2/7). The bottleneck link was set to 10Mbps with a 5ms delay. The simulation runs 1000 seconds.

We simulate real competing traffic to produce congestion events. A significant amount of traffic in today's Internet is web-related traffic, which is reported to be self-similar in nature. Research in [14] shows that self-similar traffic can be modeled with ON/OFF UDP sources with output rates drawn from a heavy-tailed distribution, such as the Pareto distribution. Thus we use ON/OFF sources as competing traffic.

In our simulation, a random process that generates rate using the Pareto distribution produces the competing traffic. We have two parameters for this random process, the Pareto distribution shape parameter, and the number of sources. The shape parameter of the Pareto distribution is set to be 1.5, which is also used to simulate web traffic in other network simulator [7]. The number of simultaneous sources used to simulate the competing traffic is up to 300 in this simulation, with each source sending at 10Kbps during an ON time. We vary the number of simultaneous sources from 0 to 300 to produce different degree of congestions.

Our model for TCP is based on continuous rates. Since there is no notion of packets in the model, it can not distinguish which flow should back off based on which flow loses packets. Therefore, we require an additional component to decide which flow would experience packet losses when congestions occur. This is the cost of simplifying the system states to continuous rates and ignoring other more detailed states, such as inter-packets intervals. In general, a flow with a higher rate at the time of congestion is more likely to get packets dropped than a flow with a lower rate. We decided to use a random process with a Bernoulli distribution to control whether a flow experiences packet losses or not. The output of this Bernoulli process is either a flow experiencing congestion or not. When congestion occurs, a flow's probability of experiencing a congestion event is equal to the ratio of the flow's instant rate to the bottleneck rate.

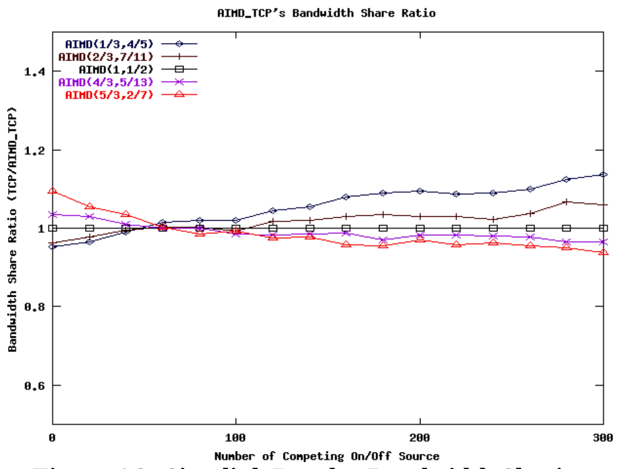

Figure 4.3: Simulink Result - Bandwidth Sharing Ratio between Normal TCP and AIMD TCP

Figure 4.3 shows our simulation results with various amounts of competing traffic. It shows the bandwidth-sharing ratio between the two competing flows. One flow is a normal AIMD TCP; another one is a TCP-friendly flow produced by an AIMD TCP with a different pair of α and β parameters. From our simulation results, the different α and β parameters (which produce different short-term rate behaviors) had an effect on the bandwidth sharing result, as did the amount of competing traffic (which affects the bottleneck link's data loss rate).

As the two flows' short-term behaviors diverge (e.g., AIMD(1/3, 4/5) differs more from AIMD(1,1/2) than AIMD(2/3, 7/11) does), the bandwidth share diverges from equal share. For similar short-term behaviors, the two flows receive close to even bandwidth share, because the two flows have similar transmission rates.

Our simulations also show that, as the amount of competing traffic increases, the bandwidth share ratio varies significantly. We plan to quantify the effect of the amount of competing traffic (which affects the data loss rate) on bandwidth share ratio.

Our model of congestion experiences could be too simple to model reality. To verify our result, we used network simulator ns2 to simulate the same TCP flows and competing traffic. Ns2 simulator generates asynchronous congestion signals directly by simulating the network in units of packets. Only flows that lose packets back off. The rest of the set up is the same as the Simulink set up.

Figure 4.4 and Figure 4.5 show our experimental results with ns2. Figure 4.4 shows the bottleneck link packet loss rate versus the number of competing sources. Figure 4.5 shows the bandwidth share ratio between a TCP flow and an AIMD flow.

As shown in Figure 4.5, ns2 produces a similar result to the Simulink simulation, although not the exact result. It shows a similar uneven bandwidth share between AIMD TCP and normal TCP, but the effect of the AIMD parameters' adjustment on the bandwidth sharing is more dramatic (uneven bandwidth share ratio ranges up to 1.5 rather than only up to 1.15 in the Simulink experiment).

Many factors contribute to the difference between these two simulations. Ns2 mimics reality closer than our Simulink model and thus introduces more random events. One significant difference is that TCP slow-start and time-outs are considered in the ns2 simulation, while in the Simulink model, we only account for the AIMD algorithm that models TCP's congestion control.

Even with these differences, one common result is reached — *a fixed AIMD parameter relation, such as Equation (3.2), does not guarantee bandwidth shared evenly among TCP flows*. Additional factors, such as timeout and slow-start stages, merely reinforce this result.

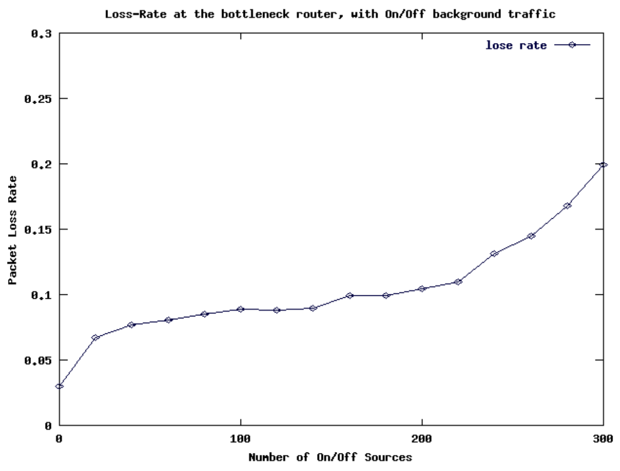

Figure 4.4: NS2 Result: Packet Loss Rate

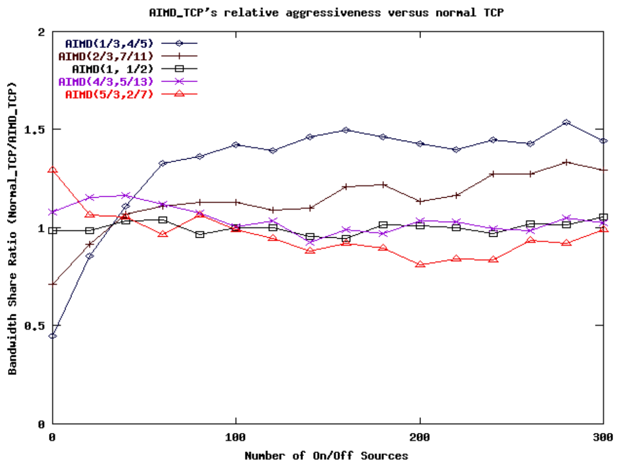

Figure 4.5: NS2 Result - Bandwidth Share Ratio Between Normal TCP and AIMD TCP

5. Factors beyond TCP-friendliness

From our experimental results, the condition currently used to judge TCP-friendliness of flows is not adequate to determine which congestion controls will share network bandwidth fairly with one another and with TCP. That condition was the basis for AIMD-based "TCP-friendly" protocols (based on an algebraic relation between AIMD parameters α and β). That raises the question: what factors beyond TCP-friendliness have a significant impact on bandwidth sharing behavior?

Differing loss event distributions cause a congestion-controlled flow to consume a different bandwidth share than another flow. The only factor that could affect the loss process of an ordinary FCFS queue differently for two flows is their rates.

Floyd, et al., [9] proposed the notion of TCP-friendliness, which is a first order rule to keep fair bandwidth share between new flows and traditional TCP traffic, but is based only on the average of a flow's transmission rate.

Other aspects of the transmission rate related to its short-term variation behavior – such as burst and oscillation frequency – may be important factors that could 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 may be a factor affecting flow bandwidth sharing.

Oscillation frequency describes how frequently a flow varies (borrows and returns) from its average throughput, which we believe may be another factor affecting flow bandwidth sharing.

We are working to quantify the effect on bandwidth sharing caused by aspects of short-term rate variations, such as the factors listed above. Other future work includes how to control AIMD 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 to 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 and using the same model observed experimentally unfair bandwidth sharing under asynchronous backing off. We verified the observed result using ns2 simulations.

We sought 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. *Proc. ACM SIGCOMM'88*, pp. 79-88, Aug. 1988.

[2] D. Chiu and R. Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks ISDN Systems*, 17, 1989

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

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

[5] Jitendra Padhye, Victor Firoiu, Don Towsley and Jim Kurose. Modeling TCP throughput: a simple model and its empirical validation. *Proc. ACM SIGCOMM'98*.

[6] S. Cen, C. Pu and J. Walpole. Flow and congestion control for Internet streaming applications, *Proc. Multimedia Computing Networking*, (MMCN98), 1998

[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. *Proc. ACM SIGCOMM 2000*.

[9] Sally Floyd and Kevin Fall. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Trans. Networking*, Aug. 1999.

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

[11] Matthew Mathis, Jeffrey Semke and Jamshid Mahdavi. The macroscopic behavior of the TCP congestion avoidance algorithm. *ACM Computer Communication Review*, vol. 27 no. 3, July 1997.

[12] K. Li, M. Shor and J. Walpole. Modeling the Transient Rate Behavior of Bandwidth Sharing as a Hybrid Control System. *Oregon Graduate Institute Tech. Report CSE-01-02*. Dec. 2000.

[13] M. Shor, K. Li, J. Walpole, D. C. Steere and C. Pu. Application of control theory to modeling and analysis of computer systems. *RESCCE'2000: Japan - USA - Vietnam Workshop on Research and Education in Systems, Computation and Control Engineering*, HoChiMinh City, Vietnam , June 7-9, 2000.

[14] A. Feldmann, A. C. Gilbert, P. Huang and W. Willinger. Dynamics of IP traffic: A study of the role of variability and the impact of control. *Proc. ACM SIGCOMM'99*. Cambridge, Massachusetts, August 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.

[17] K. Lee, R. Puri, T. Kim, K. Ramchandran and V. Bharghavan. An integrated source coding and congestion control framework for video streaming in the Internet. *IEEE Infocom 2000*, Tel Aviv, Israel, Mar. 2000.