Application of Control Theory to Modeling and Analysis of Computer Systems[℘]

Molly H. Shor
shor@ece.orst.edu
Department of Electrical and Computer Engineering
Oregon State University, Corvallis, OR 97331-3211 USA

Kang Li, Jonathan Walpole, David C. Steere
kangli@cse.ogi.edu, walpole@cse.ogi.edu, steere@cse.ogi.edu
Department of Computer Science and Engineering
Oregon Graduate Institute, Box 91000, Portland, OR 97921-1000 USA

Calton Pu
calton@cc.gatech.edu
College of Computing
Georgia Institute of Technology, Atlanta, Georgia 30332-0280 USA

Abstract – Experimentally, we show that Transmission Control Protocol (TCP)'s congestion control algorithm results in dynamic behavior similar to a stable limit cycle (attractor) when data from TCP flow into a fixed-size buffer and data is removed from the buffer at a fixed service rate. This setup represents how TCP buffers packets for transmission onto the network, with the network represented by a fixed-size buffer with a fixed service rate. The closed trajectory may vary slightly from period to period due to the discrete nature of computer systems. The size of the closed trajectory is a function of the network's buffer size and the network's service rate of packets in the buffer.

INTRODUCTION

The transport-layer protocol most commonly used in computer networks to provide error-free, in-order delivery of packets is TCP (Transmission Control Protocol) [1]. TCP is a *protocol* that determines how packets are sent over a network from one computer to another. It is specified as a standard prescribed set of rules that the sender and receiver must follow. TCP is a complicated mixture of different algorithms, applied under different conditions.

TCP's correct behavior under various conditions can be verified. "Correctness" of the TCP algorithm means that packets will be delivered in order and without errors. Packets with errors or that are not delivered will be retransmitted eventually.

Besides reliable in-order delivery of packets, TCP has a number of other useful features. TCP has a flow control mechanism to guarantee that the rate at which packets are sent will not overwhelm the receiver's maximum processing rate, overflowing the receiver's buffers. It has a congestion control mechanism that guarantees that senders will "back off" the rate that they send packets if the network gets congested. Congestion control keeps buffers in the system from constantly overflowing and bounds the time packets take to go from sender to receiver. Congestion control also probes the network for bandwidth availability and increases the rate to take up any unused bandwidth. The congestion control algorithm is the topic of this paper.

Analytic models of TCP's congestion control mechanism have been developed that predict average values of various parameters. Mathis, et al. [2], use a stochastic model to derive an expression for average throughput under the assumption that data are lost in a

[℘] This research was supported in part by DARPA contracts/grants N66001-97-C-8522 and N66001-97-C-8523, by NSF grants CCR-9876217 and by Tektronix Inc., and Intel Corporation.

periodic and predictable fashion. Padhye, et al.[3,4], develop a model to approximate the throughput of TCP flows as a function of loss rate and round trip time, during the "steady-state" operation of TCP, and compare their estimates with real-life TCP traffic. Cardwell, et al. [5], extend the model in [3] by considering timeout and TCP slowstart modes. Yeom and Reddy [6] study the model in [3] in the diff-serve (differential service) architecture, and propose an improvement to TCP in the diff-serve architecture [7]. The above analytical models are useful for computing some average values of parameters for TCP operating in isolation of other "adaptive" (dynamic) mechanisms.

Motivation for dynamic modeling

Dynamic system analysis is essential when the interaction of various dynamic subsystems must be understood. Computer system designers are finding more and more reasons to develop adaptive (dynamic) mechanisms. Some motivations include network congestion control (above), a relatively new "real-rate" class of applications that require their rates to be matched to an external rate (e.g., network video, network audio, web servers), quality of service specifications for such applications, and any application that adapts its behavior to resource availability. For instance, an adaptive video player may drop packets in response to low bandwidth allocation to meet certain user-specified "Quality of Service" specifications [15].

Multiple adaptive mechanisms implemented in a single system are likely to interact. How they interact is relevant to design. Computer system researchers would be assisted in their design efforts by the development of dynamic modeling techniques that apply to their systems.

TCP is the dominant congestion control protocol that accounts for more than ninety percent of bandwidth usage on the Internet. Internet bandwidth sharing relies on all applications using congestion control mechanisms that "conform". Applications using transport protocols that do no congestion control, such as UDP, potentially can grab more bandwidth and swamp TCP-like congestion-controlled applications. The term TCP-friendly has been proposed [8] to describe applications that share network bandwidth "fairly" with TCP-based applications.

A packet scheduler, in a router or an end-host, is an operating system component that allocates the limited bandwidth of an outgoing network interface among competing flows. Packet schedulers interact with network applications [9,10]. A TCP-like flow with a complicated packet scheduler may compete for bandwidth with a different aggressiveness in terms of how fast the flow responds to bandwidth variations.

Different types of applications require different network services in terms of delay, throughput, and reliability. Packet schedulers in networks play a major role the way bandwidth shared among flows, which determines the service and services isolations to applications [11]. Complicated packet schedulers, such as WFQ [12] and RED [13], have been designed to supply service-differential mechanisms for applications, so that different applications can be provided different levels of service. Some packet schedulers, such as [10], allocate bandwidth according to the applications' requirements.

The existing TCP models [2,3,4,5,6] focus on how much bandwidth a "TCP-friendly" application gets for a certain loss rate. Throughput by itself does not show how TCP reacts to bandwidth changes. It doesn't answer the following questions: (1) Do TCP-friendly applications reach a new stable equilibrium or periodic behavior when bandwidth changes? (2) How fast does TCP respond to bandwidth allocation variations? If an application responds to bandwidth changes slower than another application, then one application may get more share of bandwidth, at least temporarily, even if they get same average throughput under the same loss rate. The models above do not address TCP's dynamic behavior in terms of its stability and responsiveness, which we believe are important factors related to bandwidth sharing and are necessary to understand its interactions with other adaptive mechanisms.

Contribution of paper

In this paper, we describe the dynamic behavior of TCP's congestion control algorithm, using phase portraits plotted in state space, based on a simple experiment. These models were developed for fixed service rate and fixed buffer size. The dynamic behavior was examined for various service rates and various buffer sizes to determine how those variables affected the state trajectory.

TCP's congestion control algorithm results in dynamic behavior similar to a stable limit cycle

(attractor). The closed trajectory may vary slightly from period to period due to the discrete nature of computer systems. The size of the closed trajectory is a function of the service rate and buffer size.

TCP CONGESTION CONTROL

TCP is the dominant transport layer protocol in the Internet today [14]. Most Internet applications, such as web browsing and file transfer, are built using TCP. TCP is an adaptive protocol. It adjusts the sender's data output rate according to the network path characteristics and the receiver side behaviors.

TCP contains several control components. This text focuses only on a major one called TCP congestion control, which controls the sender's packet transmission rate based on whether the network path is congested or not. The widely deployed Reno-style TCP includes at least four congestion control modes (slow-start, congestion avoidance, fast recovery and fast retransmission). TCP switches control modes when special events happen, such as timeout (when no acknowledgment is received for a packet after a certain time) and resuming from a long idle time. The dominant congestion avoidance controller is studied here.

We can represent TCP congestion control as a feedback control system that outputs a signal onto the network that probes the network state, which is then used to control the data output rate. This feedback control system is illustrated in Figure 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.

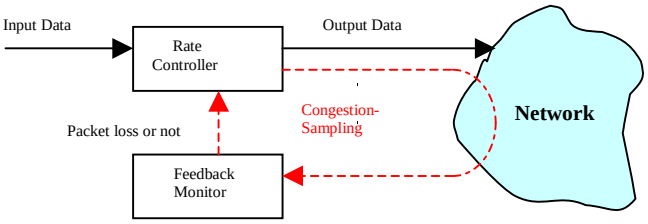

Figure 1. TCP Congestion Control System

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. If the acknowledgment for a packet arrives, the result is 1. Otherwise, the packet is assumed lost, and the result is 0. TCP detects this packet loss by looking at out-of-order acknowledgements. If acknowledgements 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 *additive increase*, *multiplicative decrease* 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 one; otherwise TCP decreases its congestion window by half.

EXPERIMENT ENVIRONMENT

In order to study the behavior of TCP congestion control, we set up the following environment, which can be easily controlled by us.

Assume that a network path's characteristic can be modeled roughly as a leaky bucket. A leaky bucket is a buffer with two specified parameters, the bucket size and the leak rate. The bucket size simulates a network path's buffering capacity. The leak rate simulates a network path's servicing rate.

We run a simple TCP application, and we inject a leaky bucket simulator between TCP and the network (Figure 2). From TCP's point of view, the leaky bucket controller is just a network link inside the

network. By adjusting the leaky bucket's leak rate and bucket size, we can produce a wide range of network path characteristics. By monitoring TCP's states under different path characteristics, we can study the behavior of TCP congestion control in a controlled environment.

bucket). The network's allocated service rate determines how quickly the leaky bucket is emptied. If TCP is successful, then the network's service rate will be matched, on average, by the outgoing data rate. The buffer fill levels determine TCP congestion

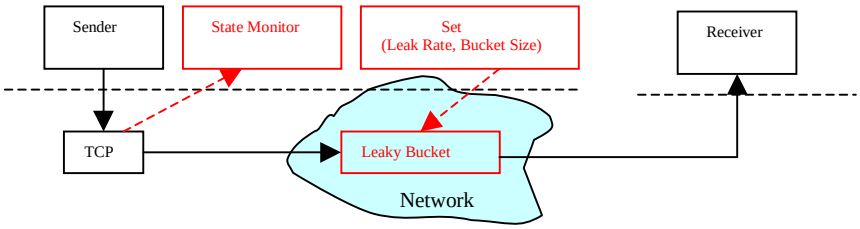

Figure 2: Experiment Environment

EXPERIMENT RESULTS

The set of variables that describe a TCP congestion control system in detail is very large. (For example, the delay interval of every two outstanding packets in the network.) It is not possible to draw them on paper.

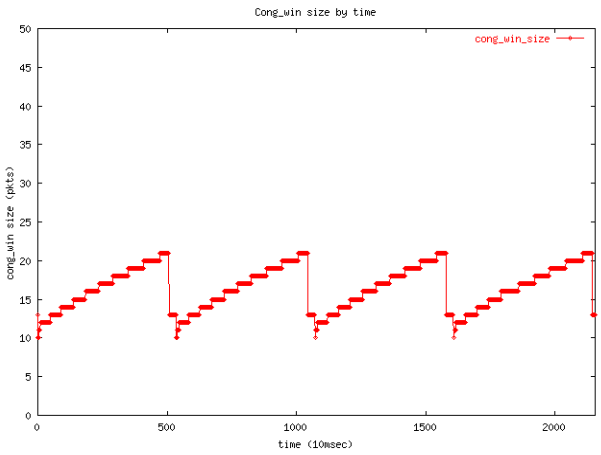

Figure 3(a). Sawtooth "steady-state" congestion window

By selecting which features are important to us, or are important in determining "what happens next" in the system, we can plot the phase portrait for a simplified model of the system.

We chose the network's (allocated) service rate, the round-trip-time, and TCP's congestion window size as the three state variables. The reasons that we choose these state variables are as follows. In TCP congestion control, the congestion window size determines the outgoing data rate (into the leaky

control's round-trip time. Thus, the round-trip time is a useful indicator of the buffer fill levels. When the buffers overflow, packets are lost and the congestion window size is reduced. The round-trip time also represents the delay in the feedback loop.

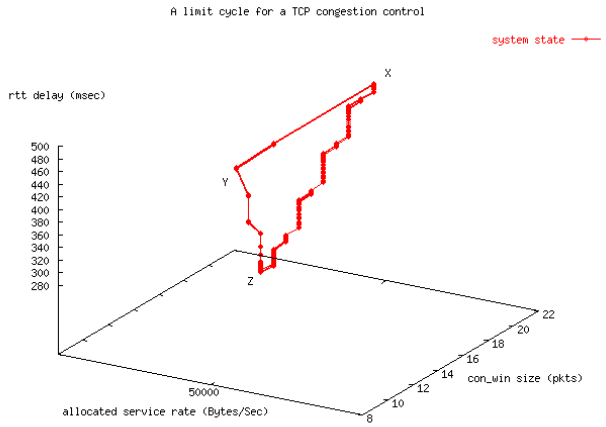

Figure 3(b). Limit cycle for TCP congestion control

Experimentally, we have found that TCP's state follows a limit cycle for a fixed service rate and network buffer capacity.

TCP congestion control results in a limit cycle

A well-known behavior of TCP congestion control is that, when it reaches it's "steady-state", its congestion window shows a saw-tooth shape (Figure 3(a)). Our experiments show that when TCP's congestion window shows the saw-tooth shape, its

trajectory is a limit cycle. Actually, the limit cycle is the result of the TCP congestion control algorithm. If we use a triangle (XYZ) to approximate the limit cycle shown in Figure 3(b), then the three edges of the triangle correspond to the three stages in TCP congestion control. Packet losses occur when the bucket is full, which also corresponds to a long RTT. Apex X indicates the system's state when a packet loss is detected. After the packet loss, the congestion window size reduces very quickly. The edge XY shows this multiplicative backoff stage.

After the backoff, TCP starts the additive rate increase. It starts with a lower rate than the network service rate, and keeps increasing until apex Z, at which TCP's rate matches the network service rate. The downward edge YZ of the triangle indicates this stage. When TCP's output rate is higher than the network service rate, packets begin to accumulate in the bucket, which causes the round-trip-time to increase. This stage is indicated by the upward edge ZX. Eventually, the bucket overflows, and packet losses occur again, again at apex X.

The limit cycle is an attractor

We manually set TCP's state to several different starting points and found that TCP's states eventually enter the same limit cycle (Figure 4). A proof is needed to show whether this is a global attractor for this service rate and buffer size.

Attractors transition

We claim that the state of TCP's congestion control follows an attractor for any given service rate and buffer size. Therefore, we believe that TCP's state will transition from one attractor to another when the service rate or the buffer's size changes. In Figure 5, we have generated the trajectories for various buffer sizes and for various service rates. Figure 5(a) shows that the size of TCP's attractor and the time TCP goes one round increase as buffer size increases. This result is consistent with the prediction from TCP models [3,4] that TCP with long RTT expands its congestion window slowly than the one with short RTT. Figure 5(b) shows that as service rate increases and buffer size equals to same amount of delay, the size of attractor increase while the time TCP goes one round is roughly unchanged. This indicates that TCPs keep expanding its congestion window in the same pace, and the TCP with large service rate can expand its congestion window size to a larger value compared to the one with small service rate. This result also conforms to the behaviors of real TCP traffic.

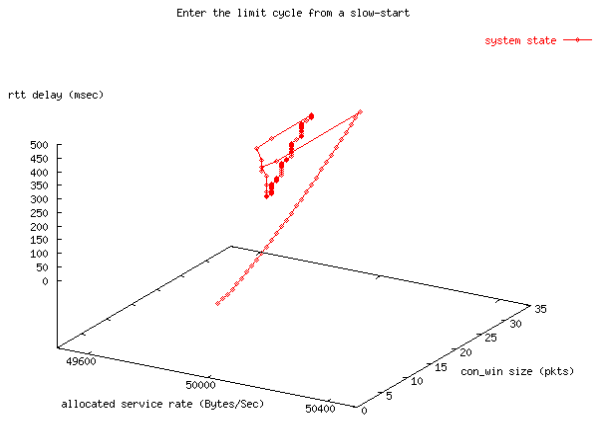

Figure 4(a). To limit cycle from "slow-start"

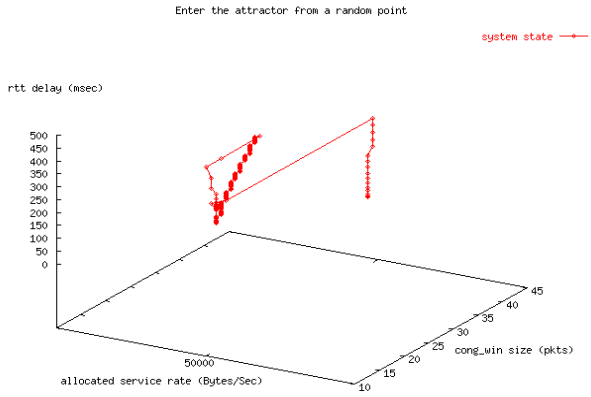

Figure 4(b). To limit cycle from random point.

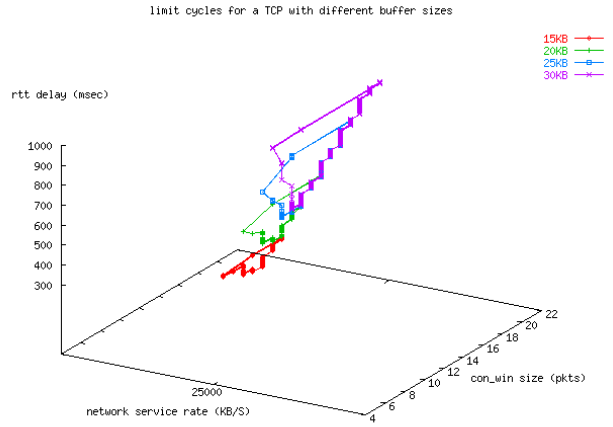

Figure 5(a). Limit cycles for different buffer sizes, from 15KB to 30KB (Smallest buffer size has lowest RTT.)

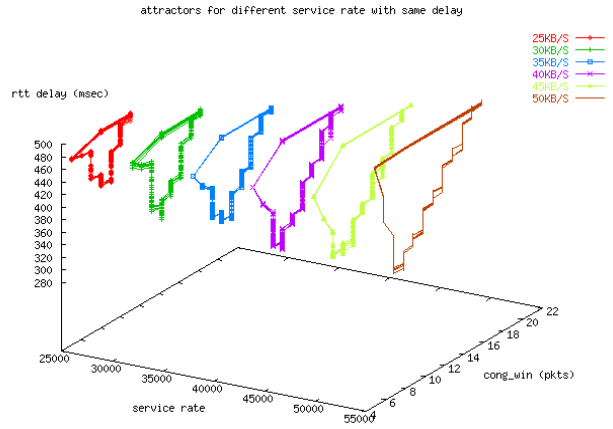

Figure 5(b). Limit cycles for different service rates, from 25KB/sec to 50 KB/sec.

APPLICATIONS OF TCP CONTROL MODEL

Predict TCP-friendliness aggressiveness from the state space model

We define the oscillation period as the time that TCP's state goes one round of its attractor. Since we sample TCP's states with a fixed interval, the number of points on one round of an attractor indicates this oscillation period value.

Network buffering (in terms of delay) affects TCP's oscillation period, as in Figure 5. Given fixed network buffering and service rate, TCP's oscillation period is determined. If we think about an arbitrary flow's oscillation period in the same condition (same service rate & network buffering), an infinitely aggressive flow can have a very small oscillation period. It would fill up the network buffer very quickly after it backs off. On the other hand, for a relative less aggressive flow than TCP will fill up the network buffer more slowly than TCP, and would have a longer oscillation period.

CONCLUSION

In this paper, we use a dynamic modeling technique, the state space model, to analyze TCP congestion control protocol. We have gained more insights to TCP congestion control just by using the dynamic model. We observe that TCP appears to have asymptotically stable periodic orbits for fixed buffer size and service rate. Some simple derivations from the model matches with results derived from other TCP models and reality. The next step of our work is to determine whether this new model can (i) predict useful properties of TCP, (ii) be used to prove how multiple adaptive mechanisms interact, and (iii) be used to design adaptive mechanisms and applications that interact with TCP in desired ways.

REFERENCES

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

[2] Matthew Mathis, Jeffrey Semke, Jamshid Mahdavi, and T. Ott. "The Macrosopic Behavior of the TCP Congestion Avoidance Algorithm". *ACM Computer Communication Review*, vol. 27, no. 3, July 1997. Also http://www.psc.edu/networking/papers/model_abstract.html

[3] Jitendra Padhye, Victor Firoiu, and Don Towsley. "A Stochastic Model of TCP Reno Congestion Avoidance and Control". CMPSCI Technical Report 99-02, University of Massachusetts, MA, 1999.

[4] Jitendra Padhye, Victor Firoiu, Don Towsley and Jim Kurose. "Modeling TCP Throughput: A Simple Model and Its Empirical Validation". *ACM SICOMM'98*. Also, CMPSCI Technical Report TR 98-008, University of Massachusetts, MA 1998.

[5] Neal Cardwell, Stefan Savage, and Thomas Anderson. "Modeling TCP Latency", *IEEE Infocom2000*. Also, http://www.cs.washington.edu/research/projects/networking/detour/

[6] Ikjun Yeom, and A.L. Narasimha Reddy. "Modeling TCP Behavior in a Differentiated Services Network".

TAMU ECE Technical Report, May 1999. Available at http://ee.tamu.edu/~reddy/papers/index.html

[7] S. Blake, D.Black, M.Carlson, E.Davies, Z.Wang and W.Weiss. “An Architecture for Differentiated Service”. RFC2475. December, 1998

[8] 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

[9] A. Aggarwal, S. Savage, T. Anderson. "Understanding the performance of TCP Pacing". *IEEE INFO-COM'2000*, 2000.

[10] Kang Li, Jonathan Walpole, Dylan McNamee, Calton Pu and David Steere. "A Rate-Matching Packet Scheduler for Real-Rate Applications". Submitted to *Computer Communications*, 2000.

[11] David D. Clark, Scott Shenker, and Lixia Zhang. "Support Real-Time Applications in an Integrated Service Packet Network: Architecture and Mechanism". *Proceeding of ACM SIGCOMM'92*, 1992.

[12] Alan Demers, Srinivasan Keshav, and Scott Shenker. "Analysis and Simulation of a Fair Queueing Algorithm". *Proceeding of ACM SIGCOMM'89*, 1989.

[13] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance", *IEEE/ACM Transactions on Networking*, vol.1, pp.397-413, August 1993.

[14] Mark Allman, Vern Paxson. “On Estimating End-to-End Network Path Properties”, *Proceeding of SIGCOMM'99*, pp. 263-274, 1999.

[15] Charles Krasic, Jonanthan Walpole, Mark Jefferys, Dylan McNamee, David Steere and Calton Pu. “Dynamic QoS Adaptation in Shared Heterogeneous Environments.” OGI-Tech-Report-CSE-99-011.