QoS Controllers for the Internet

IBRAHIM MATTA AZER BESTAVROS
COMPUTER SCIENCE DEPARTMENT
BOSTON UNIVERSITY
BOSTON, MA 02215
{matta, best}@cs.bu.edu

Abstract

In this position paper, we review basic control strategies that machines acting as *traffic controllers* could deploy in order to improve the management of Internet services. Such traffic controllers are likely to spur the widespread emergence of advanced applications, which have (so far) been hindered by the inability of the networking infrastructure to deliver on the promise of Quality-of-Service (QoS).

I. INTRODUCTION

To deliver QoS, the Internet of tomorrow is expected to implement smarter bandwidth management. Network support for QoS will eliminate the cost of over-provisioning while enhancing the commercial competitiveness of Internet service providers and carriers. We believe that one result of this trend will be the widespread emergence of machines acting as *traffic controllers*. These controllers (i) should be placed in strategic places in the Internet (e.g., in front of clients/servers or at exchange/peering points between administrative domains), (ii) should be capable of quickly inspecting and classifying packets as they go by (e.g., marking packets into precedence classes), and (iii) should control the transmission of these packets (e.g., by pacing them) to ensure quality for their applications (e.g., video streaming).

In this position paper, we identify basic QoS control capabilities that these controllers could implement. These capabilities include:

(1) congestion control for collections of flows that share the same bottleneck. Unlike traditional congestion control, "congestion-equivalent" flows are identified and managed as a set;
(2) routing flow aggregates with divergent characteristics on separate paths. Unlike traditional routing, routing metrics would respect burstiness measures, such as self-similarity and traffic correlation;
(3) hiding short-term performance degradation (e.g., loss, delay jitter) from longer-term (end-to-end) control mechanisms. Unlike traditional ad-hoc proxy approaches, the length and characteristics of the control loops that get formed between the traffic controller and the end-systems have to be taken into consideration.

Using the above functionalities, traffic controllers could increase flow throughput, reduce flow jitter and response time, and improve the stability, utilization and scalability of the network. Figure 1 shows the general architecture of a traffic controller, and the associated basic (identification and control) problems. In Section II we review four basic control schemes that could be deployed in traffic controllers. We present results from pilot studies to demonstrate the utility of these schemes. We then discuss our ongoing efforts toward a control theoretic framework in Section III, and a flexible prototype implementation in Section IV.

II. CONTROL STRATEGIES

In this section, we discuss four basic control schemes that could be depoyed by traffic controllers to ensure network stability, satisfy QoS requirements, and improve fairness across flows.

A. Aggregate Control

An important functionality we believe traffic controllers should implement is the ability to control congestion for collections of TCP flows that share the same bottleneck (instead of controlling each TCP flow individually). This improves stability, throughput, and fairness among flows. To illustrate, consider a detailed analytical model of TCP. Qiu, Zhang and Keshav [23] recently studied the aggregate performance of TCP flows, and empirically (by simulation) obtained expressions for the loss probability as a function of the number of flows and topology parameters. Given

This work was supported in part by NSF grants CAREER ANI-9701988, ESS CCR-9706685, and MRI EIA-9871022.

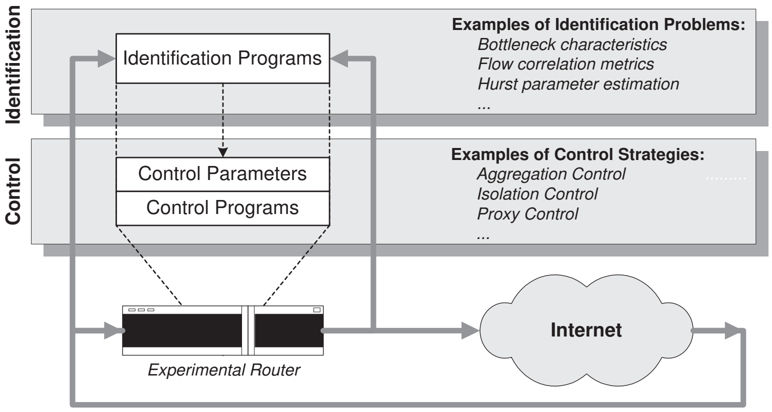


Fig. 1. General architecture of a traffic controller and basic identification and control problems.

this overall loss probability, we can use the analysis of Cardwell, Savage and Anderson [7] to obtain the time it takes to transfer a certain number of packets. Using this methodology, for 500 flows individually TCP-controlled over a T1-link[1], the loss probability on the shared link is 0.22. To transfer 50 packets (about 25KB), it takes 3.3 seconds. If we consider 50 groups of 10 flows each, and we perform TCP control on each group, then the loss probability on the link shared by these 50 super-flows is 0.038. Then, under aggregate control, it takes only 0.35 seconds to transfer 50 packets—*an order of magnitude saving in transmission time!*

Using the UCB/LBNL/VINT Network Simulator—*ns* [29], we have obtained initial simulation results [15] that underline the benefits of flow aggregation. Figure 2 shows the performance of 1MB-transfers over 32 concurrent TCP flows going through the same bottleneck link (with no cross traffic), when the 32 flows are controlled independently and in an aggregate manner. The results clearly show the advantage of flow aggregation, especially under severe congestion conditions. Most notable is the improvement in utilization under severe congestion conditions and the more even transfer times (as evidenced by the smaller ratio of maximum to mean transfer times) when flows are aggregated.

Thus, unlike traditional congestion control, traffic controllers will have to identify a set of flows sharing a common bottleneck based on measures of relationship (such as cross-correlation and cross-covariance) and manage them as a set in order to achieve better fairness, stability and throughput.

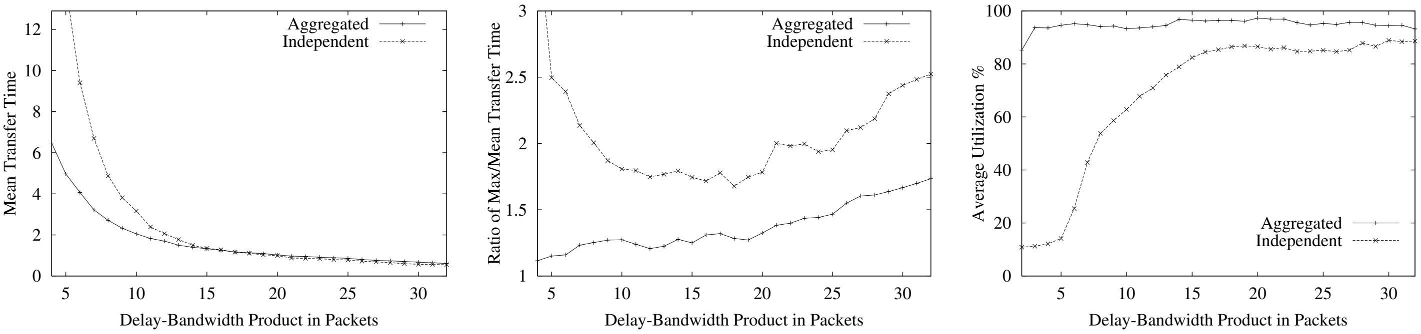


Fig. 2. ns Simulation results showing mean transfer times (left), ratio of maximum/mean transfer times (middle), and network utilization (right) for 1MB-transfers over 32 concurrent flows traversing a common congestion link. Results are shown over an increasing delay-bandwidth product.

B. Size-Based Isolation Control

Another functionality that traffic controllers could implement is routing aggregates of packet flows with divergent characteristics on separate communication paths.

Recent measurements of Internet traffic [26] show that the length (in terms of both the lifetime and the size) of TCP

[1] We assume here a buffer size of 160 packets, round trip delay of 100 msec, and packet size of 500 bytes.

flows follows a heavy-tailed distribution, i.e. only a small percentage (e.g. less than 20%) of flows are long-lived (e.g. more than 50 packets), but they carry a large percentage (e.g. 85%) of the total traffic (or bytes). Recently [9], it has been shown that when the job size follows such a distribution, a size-based task assignment policy outperforms other load balancing schemes in terms of the average slowdown (a measurement reflecting the fairness of the service) and the mean waiting time. This suggests that it may be beneficial to treat network flows differently based on their size.

Isolating Burstiness of Short Flows: Long flows behave very differently compared to short flows. First, some measurement studies [12], [28] suggest that long-lived flows have a less bursty arrival process than short-lived flows. Bursty traffic can produce volatile network congestion state, which in turn may affect the transmission of other well-behaving (less bursty and stable) long-lived flows, especially those which are using a flow/congestion control scheme (e.g. TCP connections). We expect that performance can be generally improved if the network manages well the few long-lived flows which are carrying most of the bytes.

Figure 3 shows pilot simulations [14] using the *ns* simulator [29]. We consider UDP connections (flows) of different sizes. We assume the length of each flow follows a Pareto distribution with shape 1.25, and average flow length of 300 seconds. We assume that the inter-arrival time of flows also follows a Pareto distribution with shape 1.2, and we model each active flow as a constant-bit-rate traffic. We compare three traffic management schemes employed at a traffic controller distributing flows over two parallel paths: (1) incoming flows are randomly distributed with equal probability over the two paths (i.e. a load-balanced strategy); (2) short flows[2] are routed on one path (Path 1) and long flows on the other (Path 2); and (3) each flow is first routed on Path 1, then if the flow is still active after some time threshold[3], that flow is considered long and is routed on Path 2. Table I shows the average utilization on each path, together with deviation and Hurst (self-similarity) measures [30].

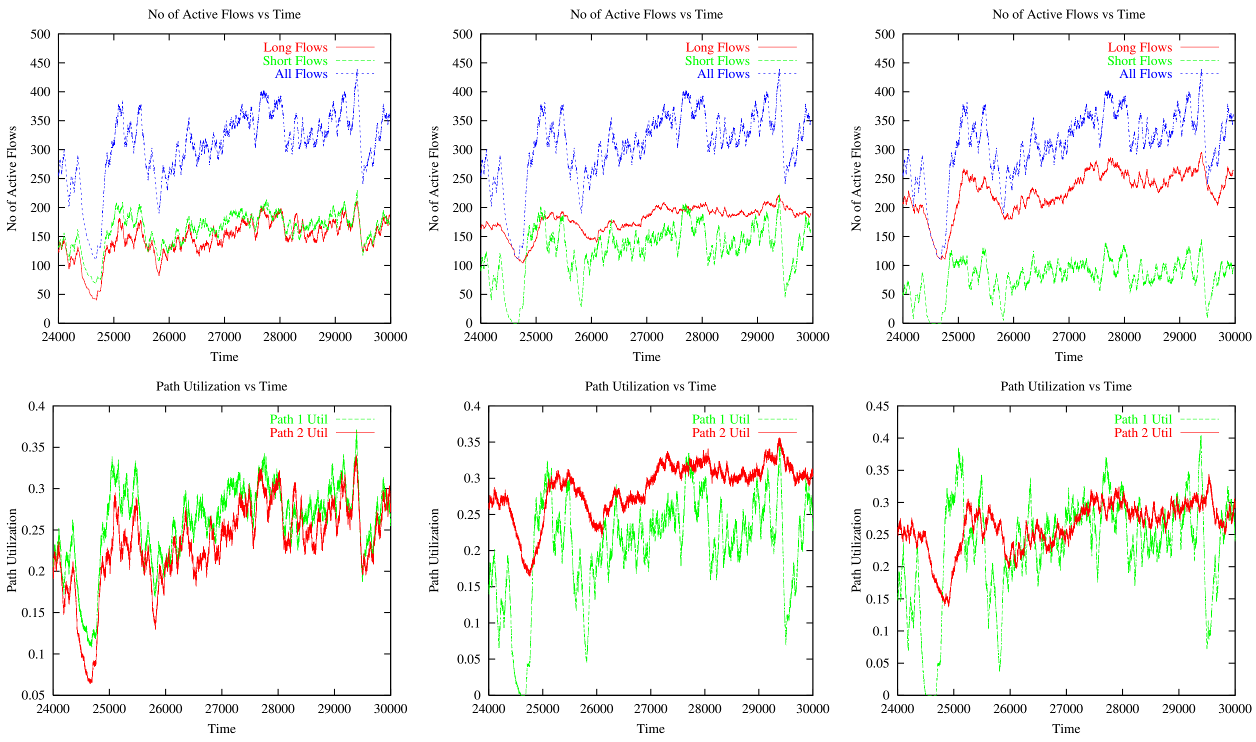


Fig. 3. Instantaneous # of active flows (top) and path utilization (bottom) for a load-balanced assignment policy of flows (left), an assignment policy that assigns short flows to one path and long flows to the other (middle), and an assignment policy that moves a flow from one path to the other once a flow length threshold is crossed.

These pilot experiments suggest that the arrival/departure process for long flows is less bursty than short flows; this is consistent with [12]. Therefore, we expect that splitting long and short flows will reduce the load variation on the route

[2]We simply assume that a flow is short if its length is less than the average.
[3]We define the threshold to be half of the average flow length.

Distribution Strategy	Path 1 average	Path 1 deviation	Path 1 H	Path 2 average	Path 2 deviation	Path 2 H
Load Balancing	0.23089	0.07194	[0.676, 0.716]	0.20789	0.07251	[0.678, 0.718]
Size-Based Splitting	0.17765	0.09144	[0.747, 0.787]	0.26113	0.06290	[0.508, 0.548]
Threshold-Based Splitting	0.19679	0.10407	[0.749, 0.788]	0.24199	0.05914	[0.576, 0.616]

TABLE I

UDP experiments: Effect of bursty arrivals/departures of short flows. Under a splitting strategy, long flows are routed on Path 2, while short flows are on Path 1.

taken by long flows, thus improving their throughput.

Isolating Window Dynamics of Short Flows: For TCP traffic, which contributes most of the Internet traffic today, long flows spend most of their time in *congestion avoidance* phase, while short flows mainly transmit in *slow start* phase. Generally speaking, the TCP congestion window changes more drastically during slow start than during congestion avoidance. Thus, isolating short and long flows may make long flow transmission more stable. Figure 4(left) shows TCP simulations, again using the *ns* simulator.[4] TCP connections of different sizes share a single bottleneck link. A flow that terminates is replaced by another flow of the same size. The evolution of the window sizes of three of the TCP connections is shown—tcp1 is longest and tcp3 is shortest. We observe that longer connections are the most affected by packet losses, and are prevented from sending at the maximum window size of 64 packets. Thus, we expect that throughput of long-lived TCP flows can be much improved by isolating them from short-lived TCP flows.

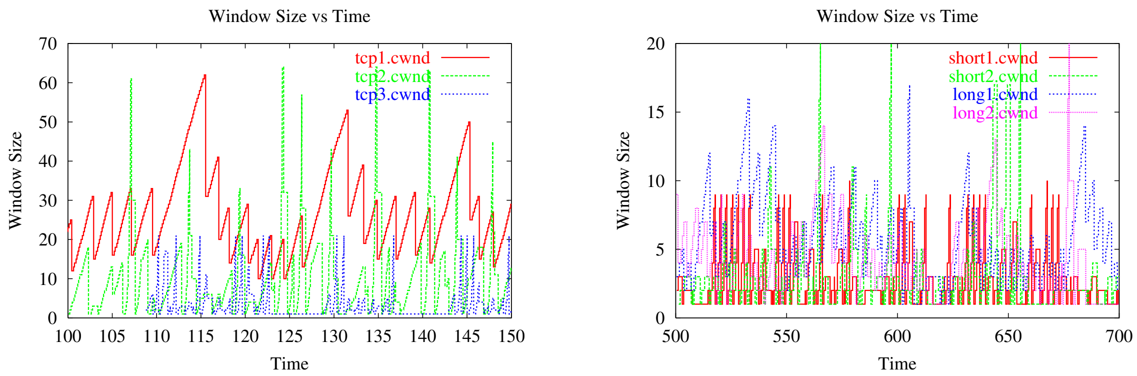


Fig. 4. Effect of packet loss on TCP window size (left) and TCP windows when short and long flows are multiplexed on the same link (right).

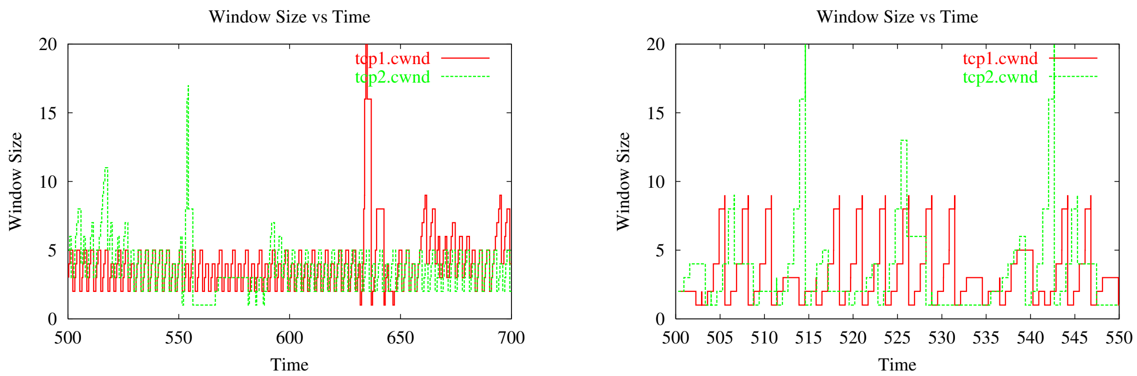
Figure 4(right) shows the evolution of the TCP window of selected flows out of 600 TCP flows: 200 long flows whose size is uniformly distributed between 900 and 1100 packets, and 400 short flows whose size is uniformly distributed between 5 and 20 packets [20]. A short (long) flow that terminates is replaced by another short (long) flow. The packet size is 576 bytes and the maximum window size is 256 packets. All 600 TCP flows share a common bottleneck 10 Mbps link with 100 msec propagation delay and a buffer size of 1000 packets. In this experiment, the throughput of long flows is observed to be around 60% of the total throughput.

Thus, in a second pilot experiment, we split the two classes of flows, giving long flows 60% of the total link resources (buffer and capacity) and short flows the remaining 40%. Figure 5(top) shows selected windows for the long class (left), and selected windows for the short class (right). We can clearly observe that performance of long flows becomes more predictable. Figure 5(bottom) illustrates that fairness is significantly improved when flows are classified, especially among long flows.[5] In particular, the 99.9^{th} percentile transmission time for long flows without isolation is 865 seconds and with isolation is 255 seconds—a 70% improvement. For short flows, the 99.9^{th} percentile transmission time without

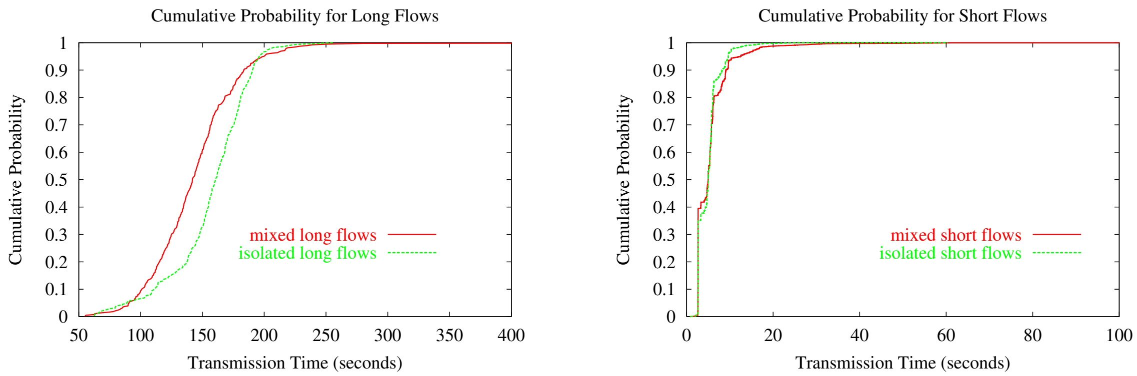
[4] All pilot simulations here are for the TCP Reno version.

[5] Note that a distribution curve that is more steep implies more fairness—a curve with a perpendicular increase implies ideal 100% fairness in the sense that all flows are experiencing the same transmission delay.

isolation is 60 seconds and with isolation is 24 seconds—a 60% improvement. Thus, unlike traditional routing, routing of incoming flows should respect burstiness measures, such as self-similarity and traffic correlation.



TCP window sizes when flows are separated into long (left) and short (right) classes.



Long (left) and short (right) file transmission time distribution—long flows of size [981:1019], and short files of size [9:11].

Fig. 5. Impact of isolation control.

We should point out that we assume a Diffserv architecture [4], whereby flows are only given *qualitative* service, and only traffic controllers located at the edge of the network keep *per-flow* information. See Figure 6 (left) for an illustration of "isolation control".

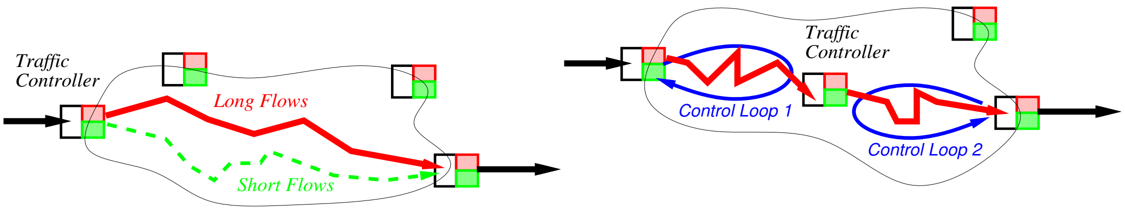


Fig. 6. Traffic controllers perform isolation control (left) and proxy control (right).

Traffic controllers will be responsible for classifying flows and marking packets as belonging to long-lived or short-lived flow. Once a flow is classified into a long flow (e.g. after a time threshold expires), traffic controllers will be able to direct the recognized flow to a new path, for example, by establishing a label-switched path using MPLS [6], [2], [27], [21], [10]. Traffic controllers have to optimally set the flow classification threshold. Furthermore, a re-routed long flow has to be gradually introduced into the ongoing group of long flows. This may require the traffic controller to purposely drop packets from a re-routed TCP flow to force it into slow-start phase.

By isolating flows based on their size, long flows will have a more stable transmission, while short flows can also benefit from improved fairness by not being shut off by long flows. Because of the highly bursty nature of short flows, we expect that such flows may not be able to make use of the bandwidth available on their path. This suggests a dynamic bandwidth allocation, whereby more bandwidth can be allocated to the paths carrying long flows. Of course, there is a strong relationship between the size threshold used to classify flows into short and long, and the bandwidth to be allocated to each class. Future work remains to study these tradeoffs and quantify fairness with respect to each class. In all pilot simulations presented in this position paper, we use TCP Reno and a tail-drop buffer management policy.[6] Future work also remains to consider other versions of TCP, such as new-Reno and SACK [11], and other buffer management policies, such as RED [13], [5].

C. QoS-Based Isolation Control

Traffic controllers should also be sensitive to service requirements. This may require isolating delay-sensitive (usually short, interactive) flows from throughput-sensitive (usually long, bulk) flows by giving them higher scheduling priority. We have shown the benefits of such isolation for best-effort (non-flow-controlled) traffic [19], [18]. Future work remains to consider flow-controlled sources (specifically TCP connections).

D. Proxy Control

An important functionality of traffic controllers is their ability to mask variability over shorter time scales to avoid disrupting control-loops operating over longer time scales. In particular, the deployment of traffic controllers as *control proxies*[7] allows a single feedback loop with a large time constant to be split into multiple, cascaded feedback loops, each with smaller (and hence manageable) time constants. See Figure 6 (right) for an illustration of "proxy control" functionality.

There are many challenging issues to be investigated for such an approach to be viable. Specifically, control proxies must be able to detect which flows may benefit from such intervention. Such a decision should be based on estimates of the length and characteristics of the control loops. Also, control proxies must not compromise the end-to-end semantics of the control mechanism. They should simply act as "low-pass filters", hiding variability at the shorter time scales, but letting through variability at longer time scales.

An example of the utility of control proxies in hiding wireless losses from TCP can be found in [25], [24].

III. A Control-Theoretic Framework for Traffic Controllers

In order to carry out stability, convergence and performance analysis of the above QoS controls, a natural tool is *control theory* [17], [22], [3]. For example, we can describe the behavior of a system of TCP flows by a discrete-time model, where the state of the system changes at discrete instants of time that correspond to the arrivals of feedback signals. The *Liapunov method* [22] provides a powerful tool to study system stability. This method allows one to obtain sufficient conditions for stability and convergence to a fixed point without actually solving the system equations. The basic idea is to find a positive-definite scalar function $V(\mathbf{S})$, where **S** is the system state, such that its forward difference $\triangle V(\mathbf{S})$ taken along a trajectory is always negative. $V(\mathbf{S})$ is said to be a Liapunov function, and is regarded as a measure of the distance of the state **S** from the fixed point. As time increases, $V(\mathbf{S})$ decreases and finally shrinks to zero, i.e. the fixed point is approached. Furthermore, the rate of convergence can be estimated from the Liapunov exponent given by $\eta = \min \left[\frac{-\triangle V(\mathbf{S})}{V(\mathbf{S})} \right]$.

The control models should take into consideration the lifetimes of flows, packet losses detected by timeouts or duplicate acknowledgments, packet generation process of various sources, both flow-controlled TCP and non-flow-controlled UDP, and the length and characteristics of the control loops that get formed between the traffic controller and the end-systems (or other controllers). The objective is to find a region of system parameters that results in the best performance, fairness and stability.

In order to exploit all possible opportunities to intelligently manage traffic, traffic controllers need to develop estimates of network properties through passive monitoring and analysis of delay and loss characteristics of individual flows [16].

[6] Some pilot simulations with TCP Tahoe also support our claims in this position paper.

[7] This is akin to having network proxy caches to scale the information retrieval process, but it is applied here to the scalability of the transport protocol.

IV. A Programmable Traffic Controller

Our future work includes the implementation of the above QoS controls in a testbed deployed in a controlled local setting as well as over the Internet. Emerging technologies, such as DiffServ and MPLS, provide the mechanisms needed for these implementations, which will be stressed by bandwidth- and QoS-demanding applications. Such a prototype should provide a programming interface to *soft* services, in which capabilities can be turned on or off and control parameters can be dynamically adjusted. For example, Lucent's Network Element for Programmable Packet Injection (NEPPI) [8] provides an ideal foundation upon which to implement the control policies we presented in this position paper. The prototype will not require any modification of existing network infrastructure, nor will it require software modification in network clients or servers.

Acknowledgments

We would like to thank Mark Crovella for various discussions and feedback on aspects of this work. Thanks also to Liang Guo and Khaled Harfoush, who performed the simulations.

References

[1] A. Aggarwal, S. Savage, and T. Anderson. Understanding the Performance of TCP Pacing. In Proc. *IEEE/INFOCOM'2000*, Tel-Aviv, Israel, March 2000.
[2] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus. Requirements for Traffic Engineering Over MPLS. RFC 2702, Network Working Group, September 1999.
[3] K-H. Becker and M. Dorfler. *Dynamical Systems and Fractals*. Cambridge University Press, 1989.
[4] Y. Bernet and et al. A Framework for Differentiated Services. Internet Draft draft-ietf-diffserv-framework-02.txt, IETF, Feburary 1999.
[5] T. Bonald, M. May, and J-C. Bolot. Analytic Evaluation of RED Performance. In Proc. *IEEE INFOCOM'2000*, Tel-Aviv, Israel, March 2000.
[6] R. Callon, P. Doolan, N. Feldman, A. Fredette, G. Swallow, and A. Viswanathan. A Framework for MPLS. Ietf draft, Network Working Group, September 1999.
[7] N. Cardwell, S. Savage, and T. Anderson. Modeling TCP Latency. In Proc. *IEEE/INFOCOM'2000*, Tel-Aviv, Israel, March 2000.
[8] A. Cohen and S. Rangarajan. A Programming Interface for Supporting IP Traffic Processing. In Proc. *First International Working Conference on Active Networks*, June 1999.
[9] M. Crovella, M. Harchol-Balter, and C. Murta. Task Assignment in a Distributed System: Improving Performance by Unbalancing Load. In Proc. *ACM Sigmetrics '98 Conference on Measurement and Modeling of Computer Systems Poster Session*, Madison, WI., June 1998.
[10] Y. Rekhter et al. Tag Switching Architecture Overview. Internet Draft, September 1996.
[11] K. Fall and S. Floyd. Simulation-based Comparisons of Tahoe, Reno and SACK TCP. *ACM Computer Communication Review*, pages 5–21, July 1996.
[12] A. Feldmann, J. Rexford, and R. Caceres. Efficient Policies for Carrying Web Traffic over Flow-Switched Networks. *IEEE/ACM Transactions on Networking*, pages 673–685, December 1998.
[13] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, pages 397–413, August 1993.
[14] Liang Guo and Ibrahim Matta. Differentiated Service Based on Flow Size. Technical Report Work In Progress, Boston University, January 2000.
[15] Khaled Harfoush and Azer Bestavros. Performance Evaluation of Aggregate TCP at Mass Servers. Technical Report Work In Progress, Boston University, June 1999.
[16] Khaled Harfoush, Azer Bestavros, and John Byers. Estimating Unicast Path Characteristics at Mass Servers. Technical Report Work In Progress, Boston University, June 1999.
[17] B. Kuo. *Automatic Control Systems*. Prentice-Hall, Inc., fourth edition, 1983.
[18] I. Matta and A.U. Shankar. Type-of-Service Routing in Dynamic Datagram Networks. In Proc. *IEEE INFOCOM*, pages 992–999, Toronto, Ontario, Canada, June 1994.
[19] I. Matta and A.U. Shankar. Type-of-Service Routing in Datagram Delivery Systems. *IEEE Journal on Selected Areas in Communications – Special Issue on the Internet*, 13(8), October 1995. Available from http://www.research.att.com/jsac.
[20] R.T. Morris. *Scalable TCP Congestion Control*. PhD thesis, Harvard University, Cambridge MA, The Divison of Engineering and Applied Sciences, January 1999.
[21] P. Newman, T. Lyon, and G. Minshall. Flow Labelled IP: A Connectionless Approach to ATM. In Proc. *IEEE INFOCOM '96*, pages 1251–1260, San Francisco, CA, March 1996.
[22] K. Ogata. *Discrete-Time Control Systems*. Prentice-Hall, Inc., 1987.
[23] L. Qiu, Y. Zhang, and S. Keshav. On Individual and Aggregate TCP Performance. Technical Report TR99-1744, Dept. of Computer Science, Cornell Univ., May 1999.
[24] K. Ratnam and I. Matta. Effect of Local Retransmission at Wireless Access Points on the Round Trip Time Estimation of TCP. In Proc. *IEEE 31st Annual Simulation Symposium '98*, April 1998.
[25] K. Ratnam and I. Matta. WTCP: An Efficient Mechanism for Improving TCP Performance over Wireless Links. In Proc. *Third IEEE Symposium on Computer and Communications (ISCC '98)*, June 1998. Code available from http://www.cs.bu.edu/faculty/matta/software.html.
[26] A. Shaikh, J. Rexford, and K.G. Shin. Load-Sensitive Routing of Long-Lived IP Flows. In Proc. *SIGCOMM'99*, Boston, MA, September 1999.
[27] I. Stoica and H .Zhang. LIRA: An Approach for Service Differentiation in the Internet. In Proc. *NOSSDAV'98*, London, UK, July 1998.
[28] K. Thompson, G.J. Miller, and R. Wilder. Wide-Area Internet Traffic Patterns and Characteristics. *IEEE/ACM Transactions on Networking*, pages 10–23, November 1997.
[29] UCB, LBNL, and VINT. Network Simulator - ns (version 2). http://www-mash.cs.berkeley.edu/ns/, 1999.
[30] D. Veitch and P. Abry. A Wavelet Based Joint Estimatior of the Parameters of Long-Range Dependence. *IEEE Transactions on Information Theory*, April 1999.