Providing Soft Bandwidth Guarantees Using Elastic TCP-based Tunnels *

Mina Guirguis Azer Bestavros Ibrahim Matta Niky Riga Gali Diamant Yuting Zhang

Computer Science Department
Boston University
Boston, MA 02215, USA
{msg,best,matta,inki,gali,danazh}@cs.bu.edu

Abstract

The best-effort nature of the Internet poses a significant obstacle to the deployment of many applications that require guaranteed bandwidth. In this paper, we present a novel approach that enables two edge/border routers—which we call Internet Traffic Managers (ITM)—to use an adaptive number of TCP connections to set up a tunnel of desirable bandwidth between them. The number of TCP connections that comprise this tunnel is elastic in the sense that it increases/decreases in tandem with competing cross traffic to maintain a target bandwidth. An origin ITM would then schedule incoming packets from an application requiring guaranteed bandwidth over that elastic tunnel. Unlike many proposed solutions that aim to deliver soft QoS guarantees, our elastic-tunnel approach does not *require any support from core routers (as with IntServ and DiffServ); it is scalable in the sense that core routers do not have to maintain per-flow state (as with IntServ); and it is readily deployable within a single ISP or across multiple ISPs. To evaluate our approach, we develop a flow-level control-theoretic model to study the transient behavior of established elastic TCP-based tunnels. The model captures the effect of cross-traffic connections on our bandwidth allocation policies. Through extensive simulations, we confirm the effectiveness of our approach in providing soft bandwidth guarantees.*

1. Introduction

The scalability of the Internet hinges on our ability to tame the unpredictability associated with its open architecture. Significant and unpredictable changes in network dynamics (and hence performance) make it harder on applications to adequately perform and even adapt if they are designed to do so. To that end, significant efforts have been expended in order to extend the basic best-effort Internet Protocol (IP) architecture so it provides hard or soft performance guarantees (on bandwidth, delay, loss, etc.) Such performance guarantees are needed by applications sensitive to Quality-of-Service (QoS), *e.g.* real-time, video streaming and games.

The IntServ architecture [3] extends IP to provide hard performance guarantees to data flows by requiring the participation of *every* router in a per-flow resource allocation protocol. The need to keep per-flow state at every router presents significant scalability problems, which makes it quite expensive to implement. To that end, the DiffServ architecture [2] provides a solution that lies between the simple but QoS-oblivious IP, and the QoS-aware but expensive IntServ solution. DiffServ encompasses the scalable philosophy of IP [16] in pushing more functionality toward the edges leaving the core of the network as simple as possible. Nevertheless, DiffServ has not yet been successful in being widely deployed by Internet Service Providers (ISPs). One reason is that DiffServ solutions still require some support from core routers (albeit much less than that of IntServ solutions). For example, the DiffServ solution proposed in [10] requires the use and administration of a dual (weighted) Random Early Drop (RED) queue management in core routers.

In addition to the need of IntServ-based and DiffServ-based solutions for network/router support, such solutions typically assume that *all* flows going through the network are managed.[1] For example, with both IntServ and DiffServ, there are no provisions for ensuring fairness amongst unmanaged best-effort flows to effectively use *excess* bandwidth in the network. We believe this to be a main drawback of these approaches as they do not lend themselves to incremental deployment on a wide-scale.

*This work was supported in part by NSF grants ANI-0095988, ANI-9986397, EIA-0202067 and ITR ANI-0205294, and by grants from Sprint Labs and Motorola Labs.

[1] For instance, typical DiffServ solutions assume that all edge routers perform necessary admission control and packet classification.

Guaranteed Throughput over Best-Effort Networks: In this paper, we investigate a solution that enables the delivery of *soft bandwidth guarantees* through the use of a best-effort, QoS-oblivious networking infrastructure. Unlike both IntServ and DiffServ, our approach does not require *any* modifications to core routers and is designed in such a way so as it may co-exist with best-effort traffic.

Our approach for delivering soft bandwidth guarantees between two points, is to adaptively adjust the demand from the underlying best-effort network so as to match the requested QoS. We do so in a way that is consistent with the proper use of the network—namely, through the use of the Transmission Control Protocol (TCP) [4] for bandwidth allocation. Specifically, to maintain guaranteed bandwidth between any two points in the network, our approach calls for the establishment of an *elastic tunnel* between these points.[2] An elastic tunnel is simply a set of TCP connections between two points whose cardinality is dynamically adjusted in real-time so as to maintain a desirable target bandwidth. Typically, the end-points of this elastic tunnel would be edge routers within a single ISP, or in different ISPs; we call these edge routers *Internet Traffic Managers* (ITM). We refer to the set of TCP connections making up an ITM-to-ITM elastic tunnel as the *ITM-TCP connections* to distinguish them from user TCP connections originating and terminating at end-hosts. Figure 1 depicts the general model we consider throughout this paper.

Example Deployments: As we hinted above, elastic TCP-based tunnels could be established between ITMs within the same ISP, or between ITMs in different ISPs. Intra-ISP tunnels could be used as a mechanism to satisfy a certain Service Level Agreement (SLA) for a given customer on an existing best-effort (i.e. QoS-oblivious) network infrastructure. For example, an ISP with a standard best-effort IP infrastructure could offer its customers a service that guarantees a minimum bandwidth between specific locations (e.g., the endpoints of a Virtual Private Network (VPN) of an organization). Inter-ISP tunnels could be used as a mechanism to satisfy a desirable QoS (namely bandwidth) between two points without requiring infrastructural support from the ISPs through which such tunnels will go through (beyond simple accounting of the aggregate volume of traffic traversing the network).

Notice that for both intra-ISP and inter-ISP deployments, and since the underlying network infrastructure is assumed to be a common IP infrastructure, it is mandatory that the envisioned "elasticity" be implemented in a manner that will not trigger network mechanisms that protect against unresponsive flows (e.g., TCP unfriendly flows). In other words, to a core router, the constituent flows of an elastic tunnel must be indistinguishable from other TCP flows.

[2]Note that other performance metrics such as delay and loss can be controlled through these elastic soft-bandwidth-guaranteed tunnels.

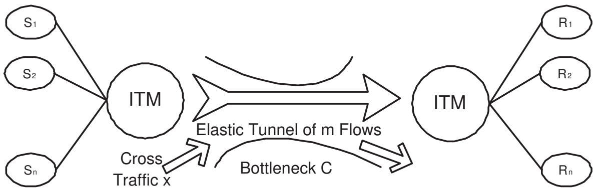


Figure 1. Elastic TCP-based Tunnel between ITMs

Without loss of generality, and for ease of presentation, in this paper we will focus on intra-ISP tunnels, with the understanding that all our results and observations are applicable to inter-ISP settings.

Paper Overview and Outline: The rest of the paper is organized as follows. In Section 2, we present our proposed architecture and its basic components. In Section 3, we present a flow-level control-theoretic model focusing on the transient behavior of our elastic TCP-based tunnels. Section 4 describes our simulations through ns-2 [6]. We revisit related work in Section 5. Section 6 concludes with a summary. Due to space limitation, we refer the reader to [9] for details on the architecture/implementation and more simulation results.

2. Overview and Architecture of ITM

Consider n regular user connections between sending and receiving end-hosts, all passing through two ITMs as depicted in Figure 1. One can think of these two ITMs as the gateways in a VPN, for example. Our main goal is to provide a soft-bandwidth-guaranteed tunnel for these user flows over an Internet path of bottleneck capacity C, which is also shared by another set of x flows, representing cross traffic. In this paper, we only consider user and cross-traffic connections to be TCP connections since TCP traffic is measured as constituting the majority of the bytes flowing over the Internet today [7]. These x cross-traffic connections present a challenge: as x keeps changing, the bandwidth allocation for the n user-TCP flows keeps changing in tandem. So an important question is whether it is possible to "counter" the change in x so as to ensure that the n user flows are able to maintain a desirable bandwidth.

Clearly without the intervention of ITMs, the answer to the above question is *no*. When different flows share a link, the effect of each individual flow (or an aggregate of flows) affects the rest since all are competing for a fixed amount of resources. However, if the ITMs dynamically maintain a number m of open TCP connections between them, they can provide a positive pressure that would equalize the pressure caused by the cross-traffic connections, if the latter occurs. Since m will be changing over time, we describe the ITM-to-ITM tunnel as *elastic*. Note that the origin ITM can decide to reduce m (i.e. relieve pressure) if x goes down—the reason is that as long as the tunnel is achieving its target bandwidth, releasing extra bandwidth should improve the performance of cross-traffic connections, which is in the spirit of best-effort networking.

To illustrate our notion of elastic tunnels and the issues involved, consider an ITM-to-ITM tunnel going through a single bottleneck link. Under normal load, the behavior of the bottleneck can be approximated by Generalized Processor Sharing (GPS) [14], i.e. each TCP connection receives the same fair share of resources. Thus, each TCP connection ends up with $\frac{C}{m+x}$ bandwidth. This, in turn, gives the m ITM-TCP flows, or collectively the elastic ITM-to-ITM tunnel, a bandwidth of $\frac{Cm}{m+x}$. As the origin ITM increases m by opening more TCP connections to the destination ITM, the tunnel can grab more bandwidth. If x increases, and the ITMs measure a tunnel's bandwidth below a target value (say B^*), then m is increased to push back cross-traffic connections. If x decreases, and the ITMs measure a tunnel's bandwidth above B^*, then m is decreased for the good of cross-traffic connections. It is important to note that the origin ITM should refrain from unnecessarily increasing m, thus achieving a tunnel's bandwidth above B^*, since an unnecessary increase in the total number of competing TCP flows reduces the share of each connection and may cause TCP flows to timeout leading to inefficiency and unfairness [13].

The main components of the origin ITM are:

Monitor: The monitor component tracks the bandwidth grabbed by the elastic TCP-based tunnel established between the origin ITM and the destination ITM. The monitor measures the bandwidth over a measurement period (MP).

Controller: The controller based on the error signal between the measured bandwidth grabbed by the m ITM-TCP flows and the desired bandwidth target B^*, adjusts the number of open ITM-TCP connections. The controller is invoked every control period (CP), which we take to be equal to MP. We discuss the performance of different types of controllers in Section 3.

Scheduler: The scheduler component is responsible for allocating the bandwidth acquired by the elastic TCP-based tunnel among the n user-TCP flows. Many scheduling policies can be used, e.g. WFQ [14]. The scheduler is called on every user packet arrival.

The ITM architecture has been implemented by an event-driven API [5]. The base of the system is an ITM kernel module, which communicates with the TCP/IP stack to retrieve packets. Preliminary implementation results could be found in [9].

3. Control-theoretic Analysis

In this section, we develop a control-theoretic model of different controllers employed at an origin ITM. Such controller determines the degree of elasticity of ITM-to-ITM TCP-based tunnels, thus it determines the transient and steady-state behavior of our soft-bandwidth-guaranteed service.

Naïve Control: This naïve controller measures the bandwidth b' grabbed by the current m' ITM-TCP connections. Then, it directly computes the quiescent number $\hat{m}$ of ITM-TCP connections that should be open as:

$$\hat{m} = \frac{B^*}{b'} m' \quad (1)$$

Clearly, this controller naïvely relies on the previously measured bandwidth b' and adapts without regard to delays in measurements and possible changes in network conditions, e.g. changes in the amount of cross traffic. We thus investigate general well-known controllers which judiciously zoom-in toward the target bandwidth value. To that end, we develop a flow-level model of the system dynamics. The change in the bandwidth grabbed $b(t)$ by the $m(t)$ ITM-TCP flows (constituting the elastic ITM-to-ITM tunnel) can be described as:

$$\dot{b}(t) = \alpha[(C - B^*)m(t) - B^* x(t)] \quad (2)$$

Thus, $b(t)$ increases with $m(t)$ and decreases as the number of cross-connections $x(t)$ increases. α is a constant that represents the degree of multiplexing of flows and we chose it to be the steady-state connection's fair share ratio of the bottleneck capacity. At steady-state, $\dot{b}(t)$ equals zero, which yields:

$$B^* = \frac{C\hat{m}}{(\hat{x} + \hat{m})} \quad (3)$$

where $\hat{m}$ and $\hat{x}$ represent the steady-state values for the number of ITM-TCP and cross-traffic flows, respectively. Based of the current bandwidth allocation $b(t)$ and the target bandwidth B^*, an error signal $e(t)$ can be obtained as:

$$e(t) = B^* - b(t) \quad (4)$$

P and PI Control: A controller would adjust $m(t)$ based on the value of $e(t)$. For a simple Proportional controller (P-type), such adjustment can be described by:

$$m(t) = K_p e(t) \quad (5)$$

P-type controllers are known to result in a non-zero steady-state error. To exactly achieve the target B^* (i.e. with zero steady-state error), a Proportional-Integral (PI-type) controller can be used:

$$m(t) = K_p e(t) + K_i \int e(t) \quad (6)$$

Figure 2 shows the block diagram of our elastic-tunnel model. In the Laplace domain, denoting the controller transfer function by $C(s)$, the output $b(s)$ is given by:

$$b(s) = \frac{C(s)G_1(s)}{1 + C(s)G_1(s)} B^*(s) + \frac{G_2(s)}{1 + C(s)G_1(s)} x(s) \quad (7)$$

where $G_1(s)$ is given by:

$$G_1(s) \quad = \quad \frac{\beta}{s} \tag{8}$$

where $\beta = \alpha(C - B^*)$. $G_2(s)$ is given by:

$$G_2(s) \quad = \quad \frac{-\alpha B^*}{s} \tag{9}$$

For the P-controller, from Equation (5), $C(s)$ is simply K_p. For the PI-controller, from Equation (6), $C(s)$ equals $K_p + \frac{K_i}{s}$. Thus, the transfer function $\frac{b(s)}{B^*}$ in the presence of a P-controller is given by:

$$\frac{b(s)}{B^*} \quad = \quad \frac{K_p \beta}{s + K_p \beta} \tag{10}$$

The system with P-controller is always stable since the root of the characteristic equation (i.e. the denominator of the transfer function) is negative, given by $-K_p\beta$. In the presence of a PI-controller, the transfer function $\frac{b(s)}{B^*}$ is given by:

$$\frac{b(s)}{B^*} \quad = \quad \frac{K_p \beta s + K_i \beta}{s^2 + K_p \beta s + K_i \beta} \tag{11}$$

One can choose the PI-controller parameters K_p and K_i to achieve a certain convergence behavior to the target bandwidth B^*. In [9], we define the transient performance measures of interest and assess the system's stability when feedback delay is present.

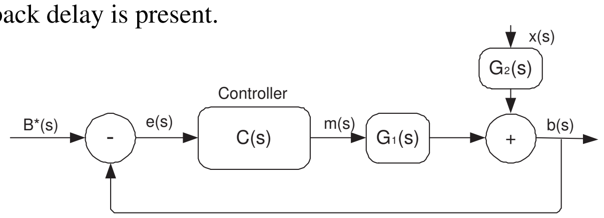
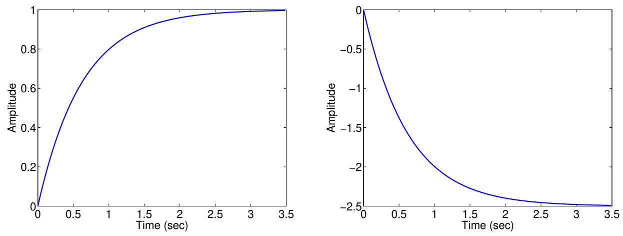


Figure 2. Block Diagram of Our Elastic-Tunnel Model

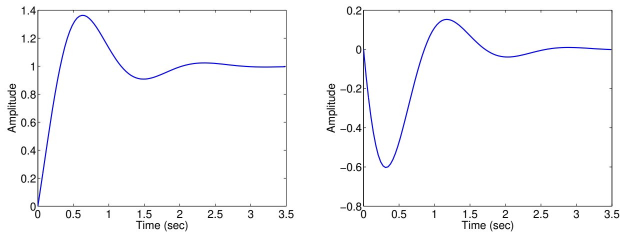
3.1. Transient Performance Results

Figure 3 shows the step response of the transfer function given in Equation (7). The left column shows the response to a step change in the target bandwidth, while the right column shows the response to a step change in the cross-traffic. Figure 3(a), for the P-controller, shows that while the response could be acceptable due to a step change in the reference bandwidth, it suffers from steady-state error (non-zero amplitude) due to a step change in the cross-traffic. Figures 3(b) and (c) show the response due to the PI-controller. One can see that through a careful choice of K_p and K_i, the transient response can be adjusted. Notice that with a PI-controller, our elastic-tunneling system can reach the target bandwidth with zero steady-state error in response to a step change in cross-traffic. Henceforth, we only focus on showing results for the PI controller. All the results of our analysis have been verified by simulations and presented in [9].



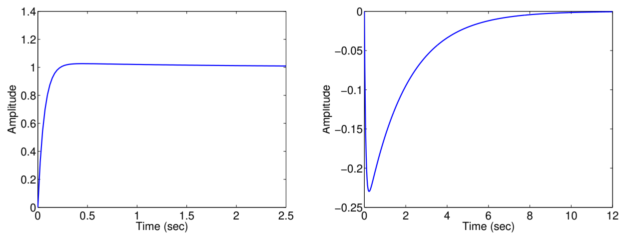
Target Bandwidth Step Response　　Cross-traffic Step Response

(a) Proportional controller with K_p = 0.1



Target Bandwidth Step Response　　Cross-traffic Step Response

(b) Proportional Integral controller with K_p = 0.2 and K_i = 1



Target Bandwidth Step Response　　Cross-traffic Step Response

(c) Proportional Integral controller with K_p = 1 and K_i = 0.5

Figure 3. Transient Analysis of our Elastic-Tunnel Model

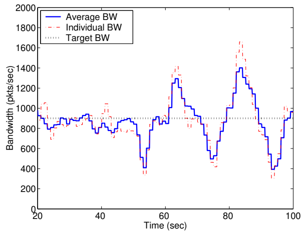
4. Simulation Results

In this section, we present results from extensive ns-2 [6] simulation experiments. These results confirm our analysis and demonstrate the effectiveness of our proposed architecture in establishing elastic soft-bandwidth-guaranteed tunnels.

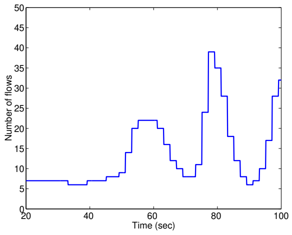
4.1. Simulation Experiments

Topology Setup: Figure 1 depicts the topology under consideration. The bottleneck link has 16Mb/s (2000 $\frac{pkts}{sec}$)

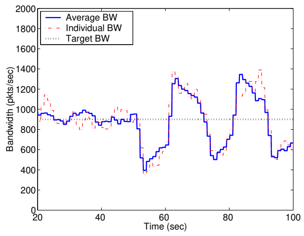
capacity and a 2-ms one-way propagation delay. We vary the propagation delay on the access links so different flows have different round-trip times. The bottleneck link is shared between ITM-TCP connections and cross-traffic connections and employs RED queue management [8].[3] All connections are considered to have unlimited data to send and they all use TCP Reno. The buffer size is chosen to be 250 packets. All packets are 1000 bytes in size. RED's minimum and maximum buffer thresholds are set to 50 and 120 packets, respectively. The RED's weight parameter was set to 0.0001 and P_{max} was set to 0.1. We focus on the transient behavior of different controllers. We ignore the first 20 seconds of the simulation time as a warm-up period. The Measurement Period (MP) as well as the Control Period (CP) are chosen to be 2 seconds.



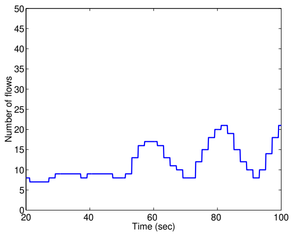
(a) Cross-traffic Square Response



(b) Cross-traffic Square Response



(c) Cross-traffic Square Response



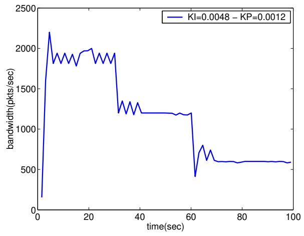
(d) Cross-traffic Square Response

Figure 4. (Top row) Effect of dynamic cross-traffic on the Naïve Controller; (Bottom row) Effect of dynamic cross-traffic on the PI Controller

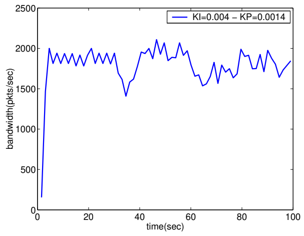
Experiment 1: The purpose of this experiment is to demonstrate the drawbacks of the naïve controller through a more dynamic behavior of cross-traffic connections. Specifically, cross-traffic connections start and stop sending data every 10 seconds starting at time 50. This has the effect of a square signal in the data sent by the cross-traffic.

[3]We note that our elastic-tunnel service does not require any specific queue management policy. Specifically, core routers may use simple FCFS (First-Come-First-Serve) queues. In practice, randomization comes from unsynchronized arrivals/departures of flows/packets and FCFS queues would serve TCP flows in a processor-sharing fashion, giving each flow its fair share of the resources.

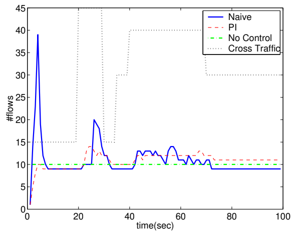
Figure 4(a) shows the behavior of the elastic tunnel under these square signals in the cross-traffic. Figure 4(b) shows the number of open ITM-TCP connections at any instant of time. As illustrated, the naïve controller fails to stabilize the number of open ITM-TCP connections which is one of our main design goals. Rather the naïve controller tends to open and close a large number of ITM-TCP connections at every control period. Figures 4(c) and (d) show the behavior of the PI-controller. One can see that it opens less ITM-TCP flows than the naïve controller. The maximum number of open ITM-TCP connections at any time was 20 as opposed to 40 connections for the naïve controller. Also, the bandwidth acquired by the elastic ITM-to-ITM tunnel tends to oscillate less than in the naïve controller case.



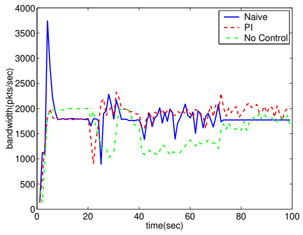
(a) Dynamic Target Bandwidth Response



(b) Dynamic Cross-traffic Response



(c) Number of open ITM-TCP flows



(d) Aggregate grabbed Bandwidth

Figure 5. Results for different controllers in a more dynamic environment

Another Setup: For the following two experiments we change the bottleneck link capacity to 50 Mb/s ($6250\frac{pkts}{sec}$). The round-trip propagation delay was chosen to be 100 msec. The Measurement Period (MP) as well as the Control Period (CP) are chosen to be 1 second. The results show that the effectiveness of our elastic-tunnel approach is insensitive to different bandwidth/delay values.

Experiment 2: In this experiment, we start with a target bandwidth of 1875 $\frac{pkts}{sec}$. At time 30, the target bandwidth decreases to 1250 $\frac{pks}{sec}$. At time 60, it is further decreased to 625 $\frac{pks}{sec}$. The cross-traffic is static. Figure 5(a) shows the bandwidth acquired by our elastic tunnel with a PI-controller, which is seen to adapt very well. We repeat the experiment, this time changing over time the number of cross-traffic flows. In particular, we start with 10 cross-traffic flows; at time 30, the number of cross-traffic flows is increased to 30; and finally at time 60, it is increased to 50. The target bandwidth is static. Figure 5(b) shows how the PI-controller stabilizes the system as expected from the analysis of Section 3.

Experiment 3: Here, we move to a more dynamic environment. In this experiment, we compare the naïve controller, the PI and the case where no control is applied. In this scenario we change the cross-traffic over time as shown in Figure 5(c). Under no control (i.e. no ITM functionality is exercised), as one would expect, the aggregate bandwidth obtained by user-TCP flows is very sensitive to changes in the cross-traffic (Figure 5(d)). The naïve controller, despite the high overshoots, finally stabilizes the system. Figure 5(c) shows the oscillating behavior of the naïve controller. It is undesirable to erratically open and close ITM-TCP connections and therefore this controller is not an optimal choice for highly dynamic environments. On the contrary, the PI-controller stabilizes the achieved bandwidth around the desired target in a less aggressive manner.

5. Related Work

In addition to QoS frameworks [3, 2] outlined in Section 1, other works have focused on developing end-system protocols that try to adapt the resources provided by the network to the needs of the application. For example, some studies (e.g., [11]) proposed different control rules for TCP behavior. By applying the right control rule, other properties can be achieved such as smoothness, aggressiveness and convergence, while maintaining friendliness to co-existing TCP traffic. This is particularly useful for streaming, real-time and gaming applications. Other studies (e.g., [12, 1, 15]) proposed that modification in transmission control rules be done on aggregates rather than individual flows, with the notion of flows sharing congestion information. For example, in [12], congestion information from a separate management connection (or using an architecture such as the Congestion Manager [1]) is used to regulate the aggregate traffic. Other techniques, such as Aggregate TCP (ATCP) [15] provides a congestion window lookup for an appropriate window size for new connections to start with. OverQoS [17] provides a controlled-loss virtual channel between overlay nodes. However, none of these techniques considered providing flows with a guaranteed bandwidth service, but rather making flows adapt to available resources more adequately.

6. Summary

We presented a framework for providing soft bandwidth-guarantees over a best-effort network. Such a guarantee is provided through the use of an elastic TCP-based tunnel running between ITMs. The target bandwidth could be dynamically adjusted to meet the needs of applications. The elasticity of the established tunnel is achieved by adjusting the number of open TCP connections between ITMs to a quiescent number, large enough to push back against cross-traffic. This is performed in a completely transparent way from the sending and receiving end-hosts. Moreover, our framework allows for the QoS support of individual applications by preferentially allocating the bandwidth provided by the established elastic tunnel. We presented simulation results showing the effectiveness of our approach in allocating the target bandwidth. Moreover, our approach remains responsive to congestion and degrades gracefully in severe congestion cases.

Acknowledgment: We would like to thank Sean Chen and Leonid Veytser for their contributions to the kernel-level implementation, and Rich West for his feedback.

References

[1] H. Balakrishnan, H. Rahul, and S. Seshan. An Integrated Congestion Management Architecture for Internet Hosts. In *Proceedings of ACM SIGCOMM*, Sep 1999.
[2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Services. *IETF RFC 2475*, Dec, 1998.
[3] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture: an Overview. *RFC 1633*, Jun 1994.
[4] V. Cerf and L. Kahn. A Protocol for packet Network Interconnections. *IEEE Transactions on Communications*, 1974.
[5] G. Diamant, L. Veyster, I. Matta, A. Bestavros, M. Guirguis, L. Guo, Y. Zhang, and S. Chen. itmBench: Generalized API for Internet Traffic Managers. *BU-TR 2003-032*, 2003.
[6] E. A. et al. UCB/LBNL/VINT Network Simulator - ns (version 2). Available at http://www.isi.edu/nsnam/ns/.
[7] S. Floyd. Measurement Studies of End-to-End Congestion Control in the Internet. http://www.icir.org/floyd/ccmeasure.html.
[8] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM TON*, Aug 1993.
[9] M. Guirguis, A. Bestavros, I. Matta, N. Riga, G. Diamant, and Y. Zhang. Providing Soft Bandwidth Guarantees using Elastic TCP-Elastic Tunnels. *BU-TR 2003-028*, 2003.
[10] L. Guo and I. Matta. The War between Mice and Elephants. In *Proceedings of IEEE ICNP*, Nov 2001.
[11] S. Jin, L. Guo, I. Matta, and A. Bestavros. A Spectrum of TCP-friendly Window-based Congestion Control Algorithms. *IEEE/ACM TON*, Jun 2003.
[12] H. Kung and S. Wang. TCP trunking: Design, implementation, and performance. In *Proceedings of IEEE ICNP*, Nov 1999.
[13] R. Morris. TCP behavior with many flows. In *Proceedings of IEEE ICNP*, Oct 1997.
[14] A. Parekh and R. Gallager. A generalized processor sharing approach to flow control in integrated services networks: the single-node case. *IEEE/ACM TON*, Jun 1993.
[15] P. Pradhan, T. Chiueh, and A. Neogi. Aggregate TCP congestion control using multiple network probing. In *Proceeding of ICDCS*, Apr 2000.
[16] J. Saltzer, D. Reed, and D. Clark. End-To-End Arguments in System Design. *ACM TOCS*, Nov 1984.
[17] L. Subramanian, I. Stoica, H. Balakrishnan, and R. Katz. OverQoS: An Overlay Based Architecture for Enhancing Internet QoS. In *Proceedings of NSDI*, Mar 2004.