

## ACC: Using Active Networking to Enhance Feedback Congestion Control Mechanisms<sup>†</sup>

*Theodore Faber*

University of Southern California/Information Sciences Institute  
4676 Admiralty Way  
Marina del Rey, CA 90292  
Phone: 310-821-1511 x190  
FAX: 310-823-6714  
faber@isi.edu

### ABSTRACT

Active Congestion Control (ACC) uses Active Networking (AN) technology to make feedback congestion control more responsive to network congestion. Current end-to-end feedback congestion control systems detect and relieve congestion only at endpoints. ACC includes programs in each data packet that tell routers how to react to congestion without incurring the round trip delay that reduces feedback's effectiveness in wide area networks. The congested router also sends the new state of the congestion control algorithm to the endpoints to ensure that the distributed state becomes consistent. We present a model for extending feedback congestion control into an Active Network, apply that model to TCP congestion control, and present simulations that show that the resulting system exhibits up to 18% better throughput than TCP under bursty traffic. In simulations without bursty traffic, the systems behaved comparably.

### 1. Introduction

Active Networking Congestion Control (ACC) is a system which uses Active Networking (AN) technology to greatly reduce the control delay that feedback congestion control systems exhibit. Each ACC packet contains either a program for a router or data used by such a program that enables a router to react to network congestion thereby

<sup>†</sup> Copyright © 1998 Institute of Electrical and Electronics Engineers. Reprinted from IEEE Network, Vol 12, no. 3, May/June 1998, pp. 61-65.

This material is posted here with permission of the IEEE. Such permission of the IEEE does not in any way imply IEEE endorsement of any of the University of Southern California's or Information Sciences Institutes's products or services. Internal or personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution must be obtained from the IEEE by sending a blank email message to [info.pub.permission@ieee.org](mailto:info.pub.permission@ieee.org).

By choosing to view this document, you agree to all provisions of the copyright laws protecting it.

This work is partly supported by the Defense Advanced Research Projects Agency through FBI contract #J-FBI-95-185 entitled Large-Scale Active Middleware. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Department of the Army, the Defense Advanced Research Projects Agency, or the U.S. Government.

avoiding the delay in communicating congestion information to endpoints.

We present simulation studies of an ACC system based on TCP congestion control mechanisms. The simulations compare the active system to standard TCP congestion control in networks with and without bursty cross traffic. In simulations without bursty traffic, the systems behaved comparably. When bursty cross-traffic is added, the active system shows as much as an 18% throughput improvement. The simulations are discussed in detail in Section 3.

Feedback-based congestion control systems do not scale well with respect to network bandwidth and delay because increases in either quantity decouple the congestion site and endpoint. The larger the end-to-end delay in a network, the longer until the endpoint can determine that the network has become congested. The higher the bandwidth of the network, the larger the amount of data the endpoint may send into a congested network in the time it takes the endpoint to detect the congestion. Bolot and Shankar have shown that under feedback-based congestion control, the duration of congestion at the bottleneck of a connection is directly related to the bandwidth-delay product[1]. High Speed Networks are an example of a class of networks with a large bandwidth-delay product.

Active Networking, the idea of reprogramming routers with data packets[2], offers an opportunity to address this shortcoming of feedback control. AN has been proposed to address many network problems, for example, to configure networks dynamically[3], or to perform application-specific tasks in the network[2]. Work by Bhattacharjee et al. has applied AN ideas to non-feedback ATM congestion control[4]. ACC applies those ideas to feedback congestion control.

ACC moves the endpoint congestion control algorithms into the network where they can immediately react to congestion. The current state of the endpoint's feedback algorithm is included in every packet. This state is small, an integer or two, to keep per-packet overhead low. When a router experiences congestion, e.g., it is forced to discard a packet, the router calculates the new window size that the endpoint would choose if it had instantly detected the congestion. The router then deletes packets that the endpoint would not have sent and informs the endpoint of its new state. Thus, ACC routers instantly *unsend* packets that would have prolonged congestion in the network. Internal network nodes beyond the congested router see the modified traffic, which has been tailored as though the endpoint had instantly reacted.

ACC reacts more quickly to congestion than a conventional feedback system can, which reduces the duration of each congestion episode. Fewer endpoints experience congestion during each episode, which improves the aggregate throughput.

ACC is designed for networks with a high-bandwidth delay product. Feedback has proven effective in networks with a small bandwidth-delay product, and ACC allows that technique to be effective in a high speed network.

The systems simulated in this paper are simple to demonstrate how ACC makes feedback control more responsive, but the reprogrammability of Active Networks enables the implementation of a wide variety of feedback controls. ACC systems can react to

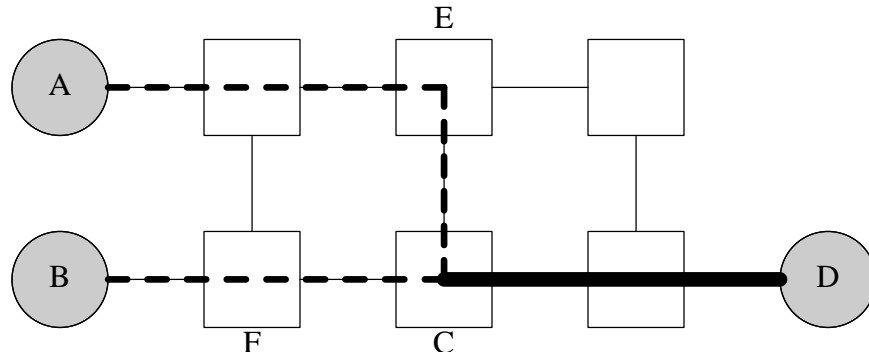
congestion using more sophisticated traffic editing than unsending packets. Rate based congestion control systems could be extended by re-spacing packets in routers closer to the congestion until the endpoint readjusts its rate. Video packets could be recoded into sequences with higher compression at routers until the endpoint adjusts. Congestion control systems could even be switched during a connection's lifetime if the nature of the traffic changes.

Section 2 of this paper describes the ACC system, and the instantiation of it based on TCP, called ACC TCP. Section 3 describes the simulation studies, and Section 4 draws conclusions and discusses future work.

## 2. The ACC System

ACC system uses AN techniques to enable router participation in both congestion detection and congestion recovery. The feedback congestion control system is extended from the endpoints into the routers. Congestion is detected at the router, which also immediately begins reacting to congestion by changing the traffic that has already entered the network.

Locating both the congestion detection and congestion response at the router removes the feedback delay; the system is stable because changes made at the routers are propagated back to the endpoints. In a conventional feedback system, congestion relief must move from the endpoint to the congestion as the endpoint sending rate is reduced; in ACC the congestion relief starts at the congested node and the change in state that sustains that relief propagates out to the endpoint.



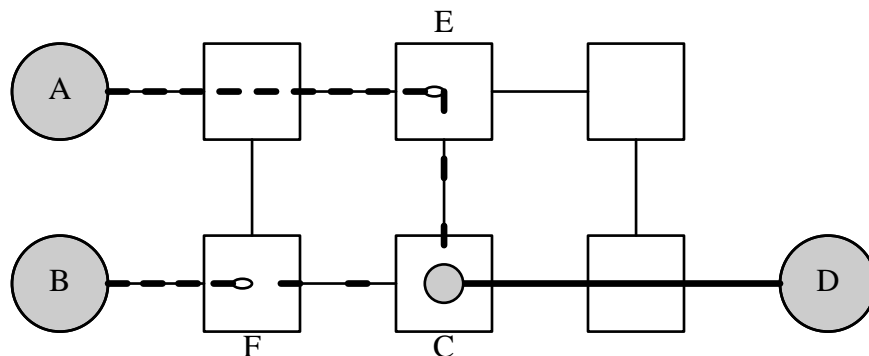
**Figure 1: A Congested Conventional Network**

A conventional feedback system will experience a delay in reacting to congestion, such as in the situation depicted in Figure 1. The packet streams from A and B cross at an internal router, C, on the way to endpoint D, congesting C. A feedback system at A or B will detect that congestion, either when it receives notification from the congested router, or when it deduces the existence of congestion because of packet loss or excessive delay. By the time endpoint A has realized C is congested, it has spent at least the propagation delay from A to C and back sending packets as though the network were uncongested, thereby making the congestion worse.

Under ACC, this delay is removed. Router C has been programmed by the first packet of the connection with instructions on how to react to congestion, and subsequent

packets include information on the current state of the endpoint's congestion control algorithm. When C detects congestion, it decides what action the endpoint would take if it had detected congestion in the state reflected by its most recent packet. The router then installs filters that delete packets that the endpoint would not have sent. These filters may be installed at the congested router's interfaces or at those of neighboring routers (E or F depending on whether congestion was detected by A or B). Finally the congested router sends a message to the endpoint telling it the new state its congestion control system. If this message is lost, the endpoint congestion mechanisms will continue to function and deduce a reasonable reaction to congestion; the loss of these messages is not a catastrophic failure.

This process is depicted in Figure 2. The small circles are the packets being sent to endpoints (A and B) to change their state. In the wake of those packets, routers E and F begin filtering out packets that have been unsent by the ACC system. The circle in router C represents the active networking component that detected congestion.



**Figure 2: An ACC network reacting to congestion**

Fewer endpoints see congestion at router C for two reasons: the congestion is relieved sooner, and the ACC system restricts the congestion reaction to fewer endpoints. The congestion is relieved because nearby routers reduce the incoming load immediately, allowing router C to serve packets and reduce its queue length. The nearby routers restrict their traffic editing to packets from the first endpoints that saw congestion. In a conventional system, those first endpoints do not immediately reduce their sending rate, which causes other endpoints to lose packets and reduce their rate.

Fewer endpoints reducing their sending rate causes the system as a whole to oscillate less, which improves the global throughput.

Moving existing congestion responses into the network is broadly applicable. Video congestion controls could recode video transmissions at the previous uncongested router, thereby providing faster response than waiting for the endpoint to reduce transmission quality. Under current systems, only the endpoint can adapt the encoding. Because congestion responses can be programmed per-flow, low priority flows, like bulk mail, may reduce their traffic very quickly in the face of congestion to allow more urgent transfers to use the scarce bandwidth.

## 2.1. ACC for TCP

As a test of the ACC principles outlined above, we have defined an active congestion control based on TCP. TCP contains a classic, well-understood feedback control system: the congestion avoidance mechanisms defined by Jacobson[5]. Endpoint sending rate is controlled by a sliding window which is advanced by packet acknowledgments. The size of the window is modulated in response to congestion along the connection's path.

The window modulation algorithm in TCP is a classic linear increase/multiplicative decrease algorithm. When congestion is detected, the window is reduced to half its current size. When a full window of consecutive packets has been acknowledged without congestion being detected, the window is increased by one maximum-sized packet. We omit the discussion of the Slow-Start algorithm because the current work considers primarily steady state effects.

An endpoint deduces that a connection's path is congested when it detects a packet loss on that connection. Such a loss can be detected by a retransmission timer expiring, or by receiving three consecutive acknowledgments of the same packet by the other endpoint.

The ACC implementation based on TCP, called ACC TCP, follows the same algorithm, except that traffic modification begins at the congested router. When a router detects a packet loss, it calculates the correct window size for the endpoint from information it provides in each packet, and forwards a packet with the new window size to the endpoint. Using TCP's window adjustment algorithm, the new window is half the old. Then the router installs a filter at the previous router on the connection's path that deletes all packets from that endpoint until it begins to act on the router's feedback (or it has detected congestion itself).

A more sophisticated ACC TCP implementation would install filters that deleted or delayed endpoint packets so that they would appear to the congested router to have been sent by a Slow-Starting endpoint. The current implementation uses the simpler filter that unsend the rest of the current endpoint's window, by discarding packets in flight. A version of ACC TCP that takes full advantage of AN to more fully edit traffic is the topic of future work.

Under ACC, the preferred method of congestion detection is notification by the congested router, but the system stills follow the TCP congestion control algorithm in the absence of that feedback. Because TCP only responds to one packet drop per round trip time, packets dropped by the filters will not cause endpoints to close their windows more quickly than they would in the face of a single packet drop.

The reaction to congestion begins at the router with the packet filter installation. This is in contrast to TCP with Explicit Congestion Notification (ECN)[6], which uses routers to notify endpoints of congestion, but applies the corrective action from the endpoint. Thus ACC will react more quickly to congestion than TCP with ECN.

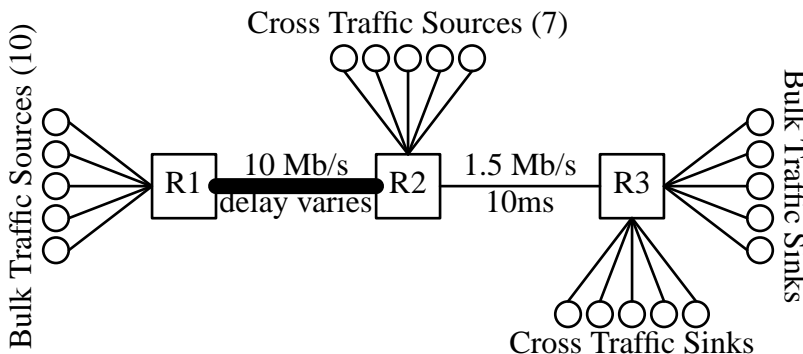
TCP Reno [7] and TCP Vegas[8] are examples of variations on TCP congestion control that we do not implement here; there are others. They differ from the basic Jacobson

TCP algorithm that we study by using different congestion detection mechanisms. We chose to extend the basic Jacobson TCP control algorithms to show that ACC can improve feedback performance. The ACC TCP dynamics are easier to see and understand in the context of the simpler algorithms. Extending our results to TCP Reno and TCP Vegas is future work.

### 3. Simulation Studies

Simulation studies of ACC TCP show that it increases endpoint's average throughput by up to 18% compared to standard TCP in the presence of bursty traffic, and provides comparable performance in a stable network. The simulations were done using links with bandwidths between 1.5 and 10 Mb/s. The bandwidth-delay product was varied by increasing the delay on an uncongested link.

All simulations were made using ns, a simulator produced by the University of California Berkeley, the Lawrence Berkeley National Labs, and the Virtual InterNet Testbed (VINT) project[9]. The simulator was extended to implement the ACC algorithms in the routers and endpoints, but not to allow full AN programmability. The modified simulator does not compile or interpret code from simulated packets.



**Figure 3: General Simulation Configuration**

Figure 3 shows the template for simulations reported in this section. The bulk traffic sources send data continuously throughout the simulation. The cross traffic sources are discussed below. All links from an endpoint to a router have a delay of 10ms and bandwidth of 10 Mb/s. All endpoints use 1000 byte packets. Each simulation is repeated for different delays on the link from the R1 to the R2. By varying that delay, the bandwidth-delay product that the bulk sources see is changed without changing the bandwidth-delay product that the cross traffic sees.

A DropTail router drops the last packet it receives when its buffer is full, like a glass overflowing; A RED router picks a random packet to discard[10]. A RED router also discards packets before it queue is full. The probability that an arriving packet causes a discard is proportional to the amount that the router's current queue length exceeds a configured minimum.

All throughputs are based on the number of useful packets received by the destination endpoint. Retransmitted packets are not considered useful.

We are primarily interested in showing that ACC supports high bandwidth-delay product networks. The simulations vary the link delay between 10 and 300 ms to increase the bandwidth-delay product. This range was selected because it is the high end of what the author perceives to be common round trip times in the Internet: the round trip time from the East coast of the US to the West is routinely 150 ms; the round trip time to a geosynchronous satellite is 250 ms. However, the important metric is the bandwidth-delay product. replacing the 10 Mb/s link with a 100 Mb/sec link and varying the delay between 1 and 30 ms in the simulations will yield similar results (modulo adjusting the buffering at routers).

### 3.1. Stable Network Simulations

These simulations show that TCP and TCP ACC behave comparably in the absence of bursty cross traffic. TCP ACC reduces the reaction time when the network changes state rapidly, but does not reduce the performance in the stable case.

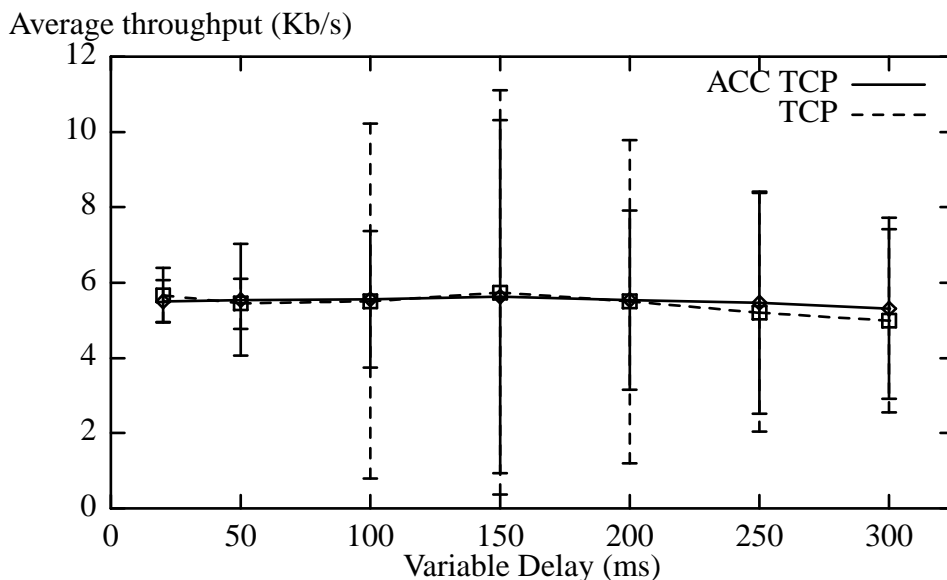
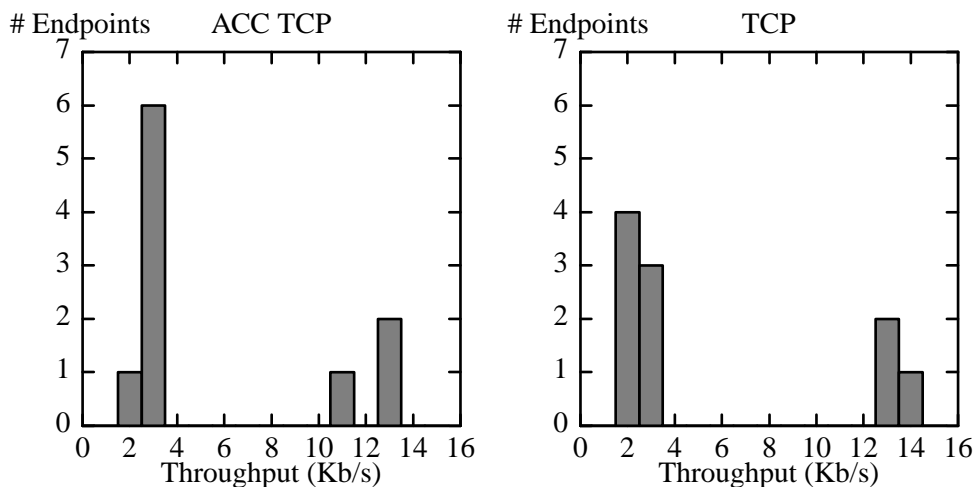


Figure 4: Throughputs with DropTail queueing (no cross traffic)

The simulations in this section use the standard simulation configuration (see Figure 3) with no cross traffic. The delay of the variable delay link ranges from 20 to 300 ms, which models the bandwidth delay products ranging from a geographically small terrestrial network to a network with a (low-bandwidth) satellite link. The goal is not to model the peculiarities of satellite networks, but to demonstrate that ACC is effective across a range of bandwidth-delay products.

The average throughput of the 10 sources is plotted in Figure 4 for various values of delay; the bars are standard deviations. Each point plotted in Figure 4 is the average throughput of an endpoint in this configuration. The routers have 25 packet buffers, and use FIFO DropTail queueing. FIFO DropTail routers queue packets first-in first-out and drop the last packet received when the queue is full.

Although the systems perform comparably in terms of bandwidth, the standard deviations are large for high-bandwidth delay product networks. The high standard deviation is due to the unfairness of TCP under DropTail FIFO queueing results in the bimodal distribution of throughputs shown in Figure 5. Figure 5 shows a histogram of the endpoint throughputs in the simulations where the variable link delay is 150ms.



**Figure 5: Histogram of endpoint throughputs for 150 ms network**

The distributions for both systems are bimodal. This is due to synchronization effects in the network caused by simultaneous losses from several endpoints. As the endpoints execute the Slow-Start algorithm[5], they are sending more packets back to back. These back to back packets are more likely to be lost, which perpetuates the cycle. The result is that some endpoints experience very few losses, and some experience many. Similar bimodal distributions observed in TCP simulation with DropTail queues[11].

The bimodal distribution of throughputs reflects the presence of two groups of endpoints, synchronized and unsynchronized. Endpoints become synchronized by simultaneous packet loss, which results in simultaneous retransmission. These endpoints enter the Slow Start phase of TCP congestion avoidance, in which two packets being sent for every acknowledgment.

Synchronized endpoints are limited by the network burst capacity, unsynchronized endpoints are limited by the network bandwidth. Packets from synchronized endpoints tend to arrive together at routers, and tend to arrive in bursts. These bursts make packet loss more likely, and reductions in window size more frequent. Unsynchronized endpoints tend to have packets evenly spaced in time. Unsynchronized endpoints lose packets less frequently because their packets tend to arrive at routers when unsynchronized endpoints are quiet, allowing them to take advantage of unused network bandwidth.

In networks with a small bandwidth-delay product, less than 100 ms in Figure 4, network burst capacity is small enough all endpoints are limited by the buffering in the routers. The network burst capacity is small enough that no source avoids burst losses long enough remain unsynchronized. The throughputs are close and the standard

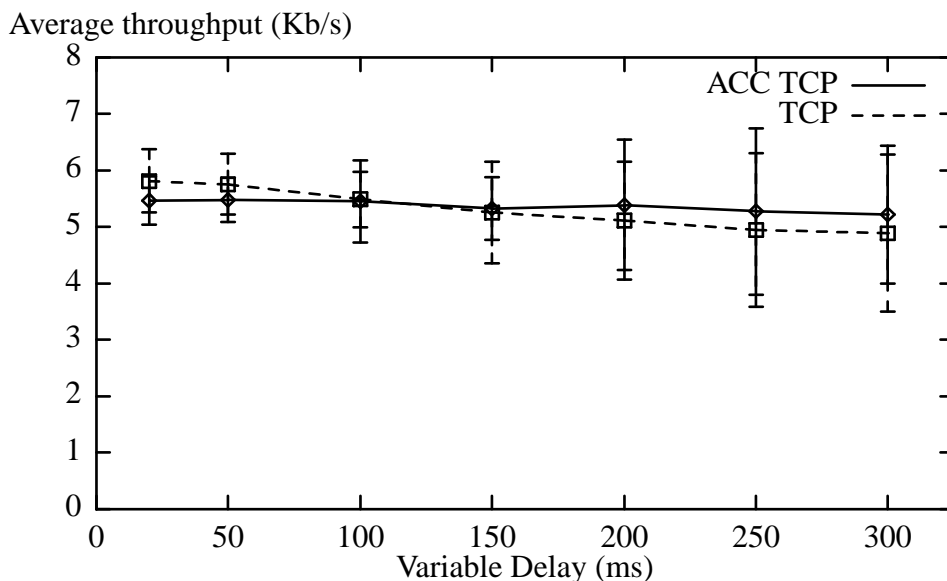
deviations small.

As the bandwidth-delay product gets larger, to the 100-150 ms range, the network burst capacity increases. Some sources avoid burst losses and remain unsynchronized. The throughputs are widely separated, and the standard deviations are large.

As the bandwidth delay gets still larger, the synchronized endpoints performance begins to improve. The larger burst capacity results in an aggregate improvement in the synchronized throughput. Unsynchronized sources now have less available bandwidth to capture. The throughputs get closer, which results in a lower standard deviation. The losses to synchronized throughput offset gains in unsynchronized throughput keeping the average nearly constant. The distribution is still bimodal.

The Random Early Detection (RED) system reduces this unfairness by dropping random packets before the queue becomes full[10]. This reduces the correlation between burstiness and packet loss, and therefore between slow-starting and packet loss. RED also provides feedback before router buffers overrun.

ACC works transparently across RED gateways, which greatly reduces this intrinsic unfairness. To demonstrate this, we repeated the simulations above using RED queuing at the routers. The results are summarized in Figure 6.



**Figure 6: Throughputs with RED queuing (no cross traffic)**

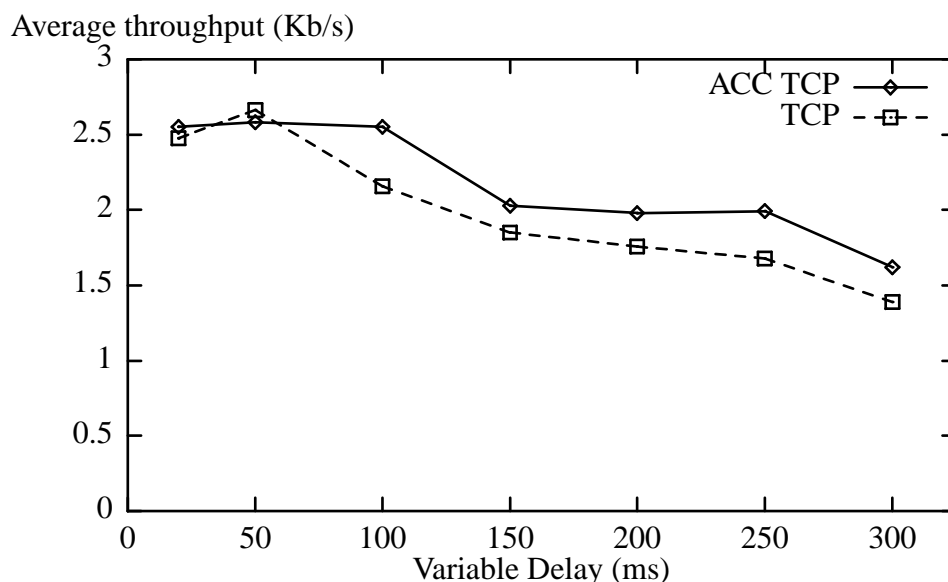
The two systems perform nearly identically under stable conditions using RED gateways. RED reduces the unfairness of the congestion controls.

ACC implies the use of RED to get reasonable fairness, but in an AN environment, this condition is easy to meet. AN already assumes considerable processing power in the router, and with the benefits and versatility RED has demonstrated[6,10], it should be implemented in AN routers.

These simulations have demonstrated that ACC TCP gives comparable performance in stable networks; the next set of simulations will show that ACC TCP performs better than TCP in a rapidly changing, high-bandwidth network.

### 3.2. Cross Traffic

These simulations demonstrate that ACC TCP reacts better than endpoint TCP to uncontrolled cross traffic. This simulation includes 10 bulk traffic sources and 7 cross traffic sources. The cross traffic sources have exponentially distributed on and off periods with means of 2.5 seconds and send at 100 Kb/s during their on periods. Cross traffic uses UDP, which does not react to congestion. Each cross traffic source sends for an average of 2.5 seconds at 100 Kb/s and then is quiet for 2.5 seconds. The goal is to model bursty cross traffic that is not responsive to congestion, for example, bursty video sources.



**Figure 7: Bulk endpoint throughputs with UDP cross traffic (RED queueing)**

As before, simulation is repeated for delays ranging from 10 to 300 ms. Figure 7 plots the average throughput seen by the bulk traffic sources against the variable delay (which directly reflects the change in bandwidth-delay product). Each point is the average of the 10 bulk transfer sources in the simulation (to make the figure more legible, standard deviations are reported in Table 1). RED queueing is used at routers.

ACC performs better than unmodified TCP by as much as 18%. Both systems' performance degrades as the bandwidth-delay product increases, because they both do depend on endpoint to router communication. Because ACC does not depend solely on that communication, it out-performs TCP. The systems show similar performance at small bandwidth-delay products because routers in the ACC system are editing traffic on behalf of the endpoints for only short intervals; the feedback loop is short enough that traditional feedback methods are effective.

In configurations with higher bandwidth-delay products, ACC TCP reduces the time of the congestion episodes and the number of affected endpoints. ACC TCP traffic backs off immediately as the bottleneck router becomes congested. As a result the congested router is indiscriminately dropping packets for a shorter time, which means that fewer endpoints see congestion. Of course, the endpoints that have their traffic filtered see a full window of congestion, but by limiting the throttling to a few hosts, the aggregate throughput remains high. RED makes it unlikely that the few endpoints that are filtered in one congestion episode are filtered in the next; a DropTail system would also show shorter congestion episodes, but at the expense of the same sources each time.

Both systems display good fairness properties as well, as shown in the standard deviations in Table 1. Table 1 also reports the improvement from using ACC TCP as a percentage of the TCP throughput for each delay value.

Delay (ms)	TCP		ACC TCP		Improvement $\left(\frac{\text{ACC TCP}}{\text{TCP}} - 1\right)$
	Mean (Kb/s)	Std. Dev. (Kb/s)	Mean (Kb/s)	Std. Dev. (Kb/s)	
20	2.47	0.684	2.55	0.366	3.2%
50	2.66	0.497	2.58	0.312	-3.1%
100	2.16	0.351	2.55	0.168	18%
150	1.85	0.218	2.03	0.187	10%
200	1.76	0.414	1.98	0.367	13%
250	1.68	0.212	1.99	0.396	18.5%
300	1.39	0.243	1.62	0.430	16.5%

**Table 1: Throughput statistics for UDP cross traffic simulation (RED queuing)**

#### 4. Conclusions and Future Work

This work has focused on describing how AN can be used to augment feedback congestion control and that there are tangible benefits to doing so. We assert that in a well implemented Active Networking architecture, having routers edit endpoint traffic inside the network can improve performance during congestion, and does not reduce performance in stable networks.

We have demonstrated that ACC systems derived from TCP show markedly better performance in simulation than pure TCP systems. The systems behave equivalently in stable networks, but the ACC system reacts faster to congestion. ACC systems are superior in maximizing throughput in networks of bursty traffic sources.

This work is preliminary in the sense that it has only shown the benefits of augmenting an existing feedback system. We intend to adapt other feedback congestion controls to the ACC model, such as Dynamic Time Windows[12] and more complicated TCP variants like Vegas. These different systems will allow us to evaluate how ACC performs relative to other congestion control methods.

We also intend to design a system based on the assumption that there are AN routers in the network to help with congestion detection and recovery. This work demonstrated

that existing protocols can benefit from extending their feedback systems into the network. A feedback system designed with a knowledge of programmability can do more aggressive control in the network. For example, the system could rearrange packets in time at internal routers, rather than discarding them. Perhaps packets could be dynamically routed around the congested router, using temporary bypasses to avoid routing oscillations. Applications to multicast congestion control may also be possible.

We also intend to implement some form of ACC on a physical Active Network to see if today's systems can support it. ACC demands little processing power from the routers, and will make a good test of AN functionality. Such an implementation will also answer questions about how feasible ACC is in today's hardware.

### Biography

Theodore received his B.S. in Computer Science from Washington University in St. Louis in 1989, and his M.S. and Ph.D in Computer Sciences from the University of Wisconsin in 1990 and 1994, respectively. He joined the University of Southern California's Information Sciences Institute in 1994 as a Computer Scientist in the Computer Networks Division. His research interests include congestion control in high speed networks, operating systems, and active networking.

### References

1. J-C Bolot and A. Shankar, "Dynamical Behavior of Rate Based Flow Control Systems," *Computer Communications Review*, vol. 20, no. 2, ACM SIGCOMM (Apr. 1990).
2. David L. Tennenhouse and David J. Wetherall, "Towards an Active Network Architecture," *Computer Communication Review*, vol. 26, no. 2, pp. 5-18, ACM SIGCOMM (April 1996).
3. Y. Yemini and S. de Silva, "Towards Programmable Networks," *IFIP/IEEE International Workshop on Distributed Systems Operations and Management*, L'Aquila, Italy (October 1996).
4. Samrat Bhattacharjee, Ken Calvert, and Ellen Zegura, "On Active Networking and Congestion," *Technical Report GIT-CC-96-02*, College of Computing, Georgia Tech, Atlanta, GA (1996).
5. Van Jacobson, "Congestion Avoidance and Control," *Proc. SIGCOMM Symposium on Communications Architectures and Protocols*, pp. 314-329, ACM SIGCOMM, Stamford, CA (Aug 16-19 1988).
6. Sally Floyd, "TCP and Explicit Congestion Notification," *ACM Computer Communication Review*, vol. 24, no. 5, pp. 10-23, ACM (October 1994).
7. Van Jacobson, "Berkeley TCP Evolution from 4.3-Tahoe to 4.3-Reno," *Proceedings of the Eighteenth Internet Engineering Task Force*, pp. 363-366, University of British Columbia, Vancouver (September 1990).
8. Lawrence S. Brakmo, Sean W. O'Malley, and L. Peterson, "TCP Vegas: New techniques for congestion detection and avoidance," *Proceedings of ACM SIGCOMM Symposium on Communications Architectures and Protocols*, pp. 24-35, ACM, London, UK (August 1994).
9. Steve McCanne, Sally Floyd, and Kevin Fall, *UCB/LBL Network Simulator NS* (1996.), available electronically from <http://www-mash.cs.berkeley.edu/ns/ns.html>.
10. Sally Floyd and Van Jacobson, "Random Early Detection gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397-413 (August 1993).
11. S. Floyd and Van Jacobson, "Traffic Phase Effects in Packet Switched Gateways," *Computer Communications Review*, vol. 21, no. 2, pp. 26-42, ACM SIGCOMM (Apr. 1991).
12. T. Faber, L. Landweber, and A. Mukherjee, "Dynamic Time Windows: Packet Admission Control with Feedback," *Proc. ACM Symposium on Communications Architectures and Protocols*, pp. 124-135, ACM, Baltimore, MD (August 17-20, 1992).

