Proceedings of the 21st International Conference on Distributed Computing Systems (ICDCS 2001).

The Effects of Inter-Packet Spacing on the Delivery of Multimedia Content*

Apu Kapadia†, Annette Feng†‡, and Wu-chun Feng‡§
akapadia@uiuc.edu, {afeng, feng}@lanl.gov

† Department of Computer Science
University of Illinois at Urbana-Champaign
1304 W. Springfield Ave.
Urbana, IL 61801

‡ Computer & Computational Sciences Research Division
Los Alamos National Laboratory
Los Alamos, NM 87545

§ Department of Computer & Information Science
The Ohio State University
Columbus, OH 43210

Abstract

Streaming multimedia content with UDP has become increasingly popular over distributed systems such as the Internet. However, because UDP does not possess any congestion-control mechanism and most best-effort traffic is served by the congestion-controlled TCP, UDP flows steal bandwidth from TCP to the point that TCP flows can starve for network resources. Furthermore, such applications may cause the Internet infrastructure to eventually suffer from congestion collapse because UDP traffic does not self-regulate itself. To address this problem, next-generation Internet routers will implement active queue-management schemes to punish malicious traffic, e.g., non-adaptive UDP flows, and to the improve the performance of congestion-controlled traffic, e.g., TCP flows. The arrival of such routers will cripple the performance of today's UDP-based multimedia applications.

So, in this paper, we introduce the notion of inter-packet spacing with control feedback to enable these UDP-based applications to perform well in the next-generation Internet while being adaptive and self-regulating. When compared with traditional UDP-based multimedia streaming, we illustrate that our counterintuitive, interpacket-spacing scheme with control feedback can reduce packet loss by 90% without adversely affecting delivered throughput.

Keywords*: network protocol, multimedia, packet spacing, rate-adjusting congestion control.*

1 Introduction

The ability of the Internet to support multimedia applications such as RealPlayer [17, 18] and Microsoft NetShow [13] will become increasingly difficult because these applications' unresponsiveness to network congestion places unfair demands on the network, particularly in light of an exponentially increasing volume of traffic. These applications generally blast UDP packets across a network at the expense of applications using TCP. Active queue-management schemes [3, 10, 5, 6, 16] for routers are being proposed to punish these non-adaptive applications by dropping the packets from their flows to ensure that well-behaved TCP applications do not starve for network resources. Consequently, the performance of multimedia applications will be crippled, thus providing the impetus for our work — an interpacket-spacing scheme with control feedback, layered on top of UDP, that can be used by multi-

*This work was supported by the U.S. Dept. of Energy through Los Alamos National Laboratory contract W-7405-ENG-36, by the U.S. Dept. of Energy LDRD-ER Grant 2000023, and the Los Alamos Computer Science Institute. Any opinions, findings, and conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the U.S. Dept. of Energy, Los Alamos National Laboratory, or the Los Alamos Computer Science Institute. This paper is LA-UR 01-0904.

media applications to achieve good performance in the presence of active queue management.

1.1 Insight

Based on our recent work in network traffic characterization [24, 7, 8], we observed significant packet loss even when the offered load was less than half of the available network bandwidth. An analysis of our *ns* [1] simulations revealed that this behavior was due to simultaneous bursts of traffic coming from client applications and overflowing the buffer space in the bottleneck router. Metaphorically, this could be viewed as what happens at a major highway interchange during rush hour where everyone wants to go home simultaneously at 5:00 p.m., thus "overflowing" the highway interchange. To avoid such a situation, some people self-regulate themselves by heading home at a different time, i.e., spacing themselves out from other people.

If we view vehicles as packets and the highway interchange as a router, then to avoid buffer overflow and enhance throughput, packets should not be blasted onto the network one after another. Instead, packets should be spaced out over time. To test this hypothesis, we ran live wide-area network (WAN) tests between Los Alamos National Laboratory (LANL), University of Illinois at Urbana-Champaign (UIUC), and Ohio State University (OSU). These tests consisted of sending UDP packets between LANL and either UIUC or OSU at different packet-spacing intervals. Figures 1 and 2 show the throughput and packet loss, respectively, of a representative test between LANL and UIUC [4]. When the packet spacing is zero, e.g., today's UDP-based multimedia-streaming applications, the throughput is 62 Mb/s but with a packet loss of almost 90%! With as little as 100 μs of spacing between packets, the throughput remains the same, but the packet loss drops all the way down to 35%. And when the packet spacing is 50 μs, the throughput is actually *higher* than when the packets are not spaced as in UDP-based multimedia streaming.

All curves from our other live WAN tests have the same general shape. That is, the throughput initially increases when the amount of packet spacing increases and then decreases exponentially as the amount of spacing increases further. The packet-loss percentage immediately decreases in an exponential manner as packet spacing increases.

1.2 Related Work

In 1997, Mahdavi and Floyd [12] informally proposed the notion of equation-based congestion control for unicast applications. While the "additive increase, multiplicative decrease" (AIMD) algorithm found in TCP backs off by cutting its sending rate in half in response to a single congestion indication, equation-based congestion control uses

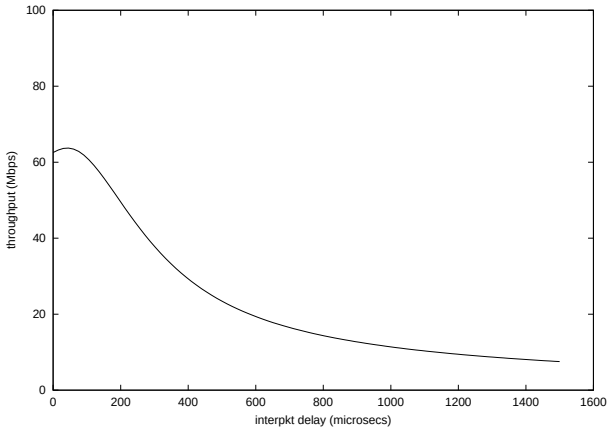

Figure 1. Delivered Throughput to the Receiver

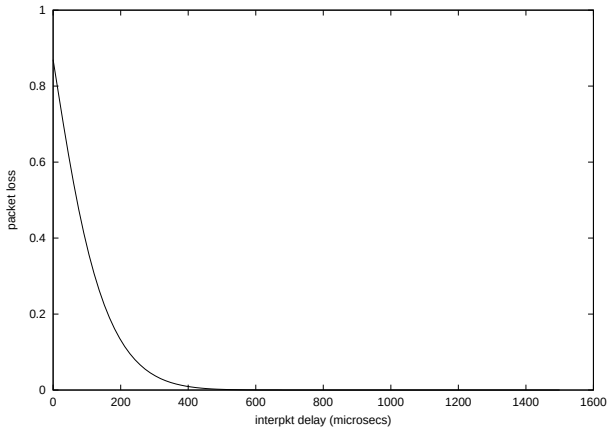

Figure 2. Packet-Loss Percentage

a control equation that more gradually and smoothly adapts its maximum rate because some real-time applications find that halving the sending rate is unnecessarily severe and can noticeably reduce the user-perceived quality [23]. Although the above work has given rise to a significant amount of research on equation-based and other types of congestion-control mechanisms [22, 20, 23, 15, 21, 9], we still do not have any deployable congestion-control mechanisms for best-effort streaming multimedia.

Previous work in packet spacing includes [11, 2]. In [11], Jain argues that rate-control protocols for congestion control may not work without the cooperation of intermediate routers because packets may get clumped together at the intermediate routers anyway. This would result in larger bursts at the intermediate routers even though the goal may have been to reduce the burstiness of the traffic. While this may have been true a decade ago, we believe that the boom

of the world-wide web and other multimedia applications creates enough interleaving traffic to maintain packet spacing between end hosts. We will substantiate this belief in Section 3.2.4.

Aggarwal et al. [2] study the effect of uniform packet spacing (or "pacing") over a round-trip time in TCP. While pacing results in better fairness, throughput, and lower drop rates in some cases, the throughput is worse than regular TCP most of the time because a paced-TCP is susceptible to synchronized losses and delays congestion notification. In contrast, we focus on the effects of packet spacing over UDP with control feedback rather than on TCP itself.

In general, our packet-spacing protocol differs from the above work in several ways. First, rather than focusing primarily on being compatible or *fair* with TCP, our rate-adjusting protocol addresses fairness while simultaneously delivering UDP-like bandwidth. Second, we accomplish the above feat by introducing the counterintuitive notion of packet spacing. Third, rather than relying on equation-based congestion control to more smoothly adapt the sending rate, we allow the sending rate to adapt as needed (based on available network resources). We then rely on transcoding, e.g., mapping a multimedia stream onto rapidly-varying available bandwidth [19], to smooth out any potentially rapid change in available bandwidth.

2 Approach

Packet spacing refers to the delay introduced between two consecutive packets, as shown in Figure 3. Here, t_s is the amount of spacing between packets, and t_x is the transmission time for each packet. By introducing such a delay, bursts of packets can be spaced out, resulting in fewer packet drops at intermediate routers and potentially *higher* throughput at the end host, as shown back in Figure 1. Thus, packet spacing can potentially be used as a mechanism to assist in congestion avoidance and control.

Based on Figure 1, the ideal operating region of our packet-spacing mechanism ranges from 50 μs to 500 μs. No packet spacing or packet spacing of less than 50 μs results in very high packet loss with less delivered bandwidth than when the packet spacing is 50 μs.

Depending on the application, the ideal packet-spacing range may be as small as 100 μs to 200 μs in order to get UDP-like bandwidth but with significantly less packet loss, e.g., at 200 μs, bandwidth is 50 Mb/s while packet loss is only 10%, or as large as 400 μs to 500 μs to obtain TCP-like reliability but with higher throughputs. To exploit this counterintuitive finding, we develop our packet-spacing protocol (PSP) to adjust the amount of packet spacing based on feedback from the network.[1]

[1] We note that at the present time, the feedback is only used for adjusting the packet spacing and that no retransmissions are done at this time.

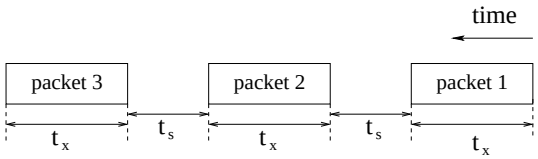

Figure 3. Packet Spacing

2.1 Packet-Spacing Protocol (PSP)

In PSP, the sender transmits packets at the highest possible rate, i.e., no inter-packet spacing, and the receiver sends acknowledgments every round-trip time (RTT) for the packets it received. (This RTT is the base propagation-delay time, not the dynamic RTT. To keep the protocol simple, we did not experiment with dynamic RTTs.)

We calculated the base RTT by performing *ping* during connection set-up.[2] After the connection is established, the sender conveys the calculated RTT to the receiver by including it within the header of each packet. Note that this is not required after the first acknowledgment is received, but we have left this provision so that dynamic RTTs can be used in the future. Each acknowledgment contains the number of packets that were received in the previous RTT.

When the sender receives such acknowledgments, it compares the number of packets sent, p_{sent}, in the previous RTT to the number of packets received, p_{rcvd}. Based on the values of p_{sent} and p_{rcvd}, the sender adapts its packet spacing ps as shown in Figure 4.

if $p_{sent} > p_{rcvd}$ (i.e., packets were lost) **then**
    /* sender must reduce its transmission rate */
    if $ps = 0$ **then**
        $ps \leftarrow 50\ \mu s$
    else
        $ps \leftarrow min(ps * 2, RTT)$
else /* sender tries to increase its sending rate */
    $ps \leftarrow ps - 2$

Figure 4. Packet-Spacing Protocol

Because our WAN experiments and simulations showed that the ideal packet spacing occurred between 0 μs and 2000 μs, we chose an initial packet spacing of 50 μs because (1) anything smaller generated significantly higher packet loss with no benefit with respect to throughput and (2) finding the ideal packet spacing within this range quickly would take no more than seven RTTs. Larger spac-

[2] A more sophisticated mechanism could be developed to get a better estimate of the RTT. However, for the purposes of our experiments, we only needed a value that was reasonable enough to provide timely feedback.

ings can be reached in only a few more RTTs because the packet spacing increases exponentially.

The $ps \leftarrow min(ps * 2, RTT)$ clause ensures that the maximum packet spacing is one RTT. This ensures that at least one packet is sent every RTT.

2.2 Damped Packet-Spacing Protocol

Due to the opposing packet-spacing decisions in PSP, our initial tests of PSP resulted in large oscillations around the ideal sending rate. To prevent this, we added the following heuristic to damp the oscillations: *If a loss occurred due to a deliberate decrease in the packet spacing (and consequently, increase in rate), then the sender reverts to the previous packet-spacing value.* Using this heuristic, the sender makes significantly smaller oscillations around the ideal operating point. Figure 5 shows a comparison between PSP and damped PSP. In this figure, each experiment was run for 100 s, and the sending rate for each was plotted. With damping, the overall throughput increased by 10%.

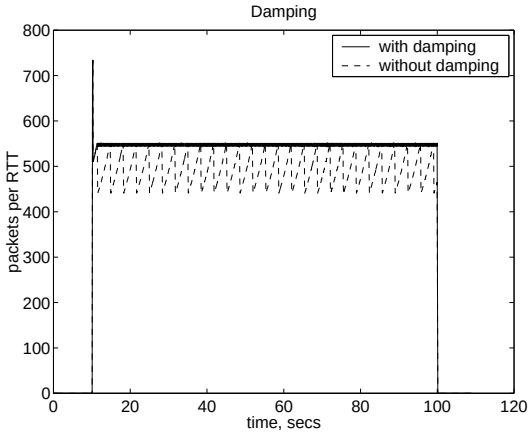

Figure 5. Oscillation Damping

3 Experiments

For our WAN simulations, we used *ns-2*, which is a network simulator developed by the VINT group [1]. We will refer to senders and receivers as *agents*, which follows naturally from the terminology employed by *ns-2*. Our simple packet-spacing agents (PSAs) implement packet spacing without feedback while our adaptive packet-spacing agents implement the damped PSP rather than the plain PSP.

3.1 Architecture

Figure 6 shows the network topology that we used in our experiments. The k nodes on the left ($n_1, n_2, \ldots, n_k$) simulate senders on an Ethernet that are transmitting via a common gateway router (e.g., LAN/WAN gateway or n_{middle}) to a WAN backbone running at 155 Mb/s or OC-3. All the receivers are aggregated into the node n_{sink}. The gateway router has a buffer size of 10 packets, 100-Mb/s Ethernet links with 2-ms delays to the senders, and a 155-Mb/s link with 40-ms delay to the receivers. This delay is typical of the delay found in a transcontinental WAN connection.

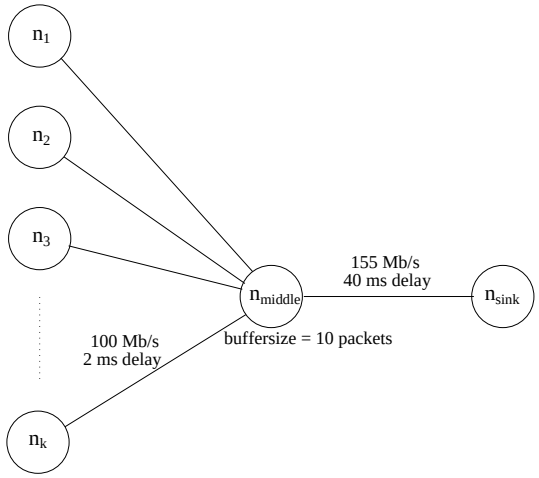

Figure 6. Topology for WAN Simulations

3.2 PSA Simulations

Here we study the behavior of (1) a single PSA with no other traffic, (2) competing PSAs, and (3) PSAs competing with TCP agents. Like Mo et al. [14] who compare TCP Reno and TCP Vegas using infinite file transfers, we use infinite file transfers for the TCP connections as well. (For the figures in this section, each data point in the simulation graphs represents the result of a 500-s simulation for a particular packet-spacing interval.)

3.2.1 Single PSA

Figure 7 shows the throughput for a single PSA for packet spacings between 0 μs and 5000 μs. (Note that there is no other traffic on the network besides that of the single PSA.) As expected, the sender and receiver throughputs are the same. This is because there is no other traffic on the channel, and therefore, no packet loss.

3.2.2 Competing PSAs

In this set of experiments, we ran simulations with 2, 4, 8, and 16 PSAs competing against each other, respectively.

Figures 8 and 9 show the results for the last case. The resulting behavior is similar to what we observed in the actual WAN experiments (i.e., Figures 1 and 2). (Note that all the 16 competing PSAs showed a similar behavior.)

In Figures 8 and 9, the region of interest occurs between 0 μs and 1000 μs. With a packet spacing of 0 μs, the sender throughput is 100 Mb/s while the receiver-realized throughput is only a measly 10 Mb/s with a packet loss of 90%! As packet spacing increases, the packet-loss percentage drops sharply, and the throughput at the receiver actually *increases* to its maximum point at 1000 μs of inter-packet spacing. This phenomena is similar to what we found with our live WAN tests in Figures 1 and 2.

3.2.3 PSAs Competing with TCP Agents

In these experiments, we ran simulations with 1, 2, 4, 8, and 16 sender/receiver TCP pairs and an equal number of PSA pairs, respectively. Figures 10 and 11 show the behavior of one particular PSA competing with 15 other PSAs and 16 TCP connections. All other simulations resulted in similar behavior. Again, we see that the behavior is strikingly similar to that seen in the actual WAN experiments. The optimal performance of the PSAs with respect to throughput and packet loss occurs at 1000 μs to 1050 μs, i.e., throughput is 11 Mb/s while packet loss is 0%.

Figures 12 and 13 show the throughput and packet-loss behavior of one particular TCP connection competing with 15 other TCPs and 16 PSAs, respectively. In these figures, we cannot help but notice that the TCP throughput does not increase beyond 2.7 Mb/s (even when the PSA throughput is low)! The reason for this behavior has nothing to do with the TCP-friendliness of our damped PSP and has everything to do with TCP's default advertised receiver window of 20 packets. This receiver's window size is the default in many operating systems and artificially limits the amount of outstanding data that a sender can have in the network. Further, the figures also show that with very small packet spacings, the PSAs operate like UDP connections (as expected), thus starving TCP connections of any bandwidth.

Figures 14 and 15 show how TCP behaves with a window large enough to keep a bandwidth-delay product's worth of information outstanding in the network. These figures show that with sufficient spacing by the PSAs, a TCP connection can consume its share of available bandwidth. For example, Figures 10 and 14 illustrate that with 5000 μs of packet spacing, each PSA receiver sees 2.23 Mb/s while each TCP receiver gets 6.57 Mb/s.

3.2.4 PSA Spacing at the Receiver

To verify our claim that packet-spaced traffic stays spaced out by the time it reaches the receiver (rather than getting clumped as claimed by [11]), we recorded the inter-arrival

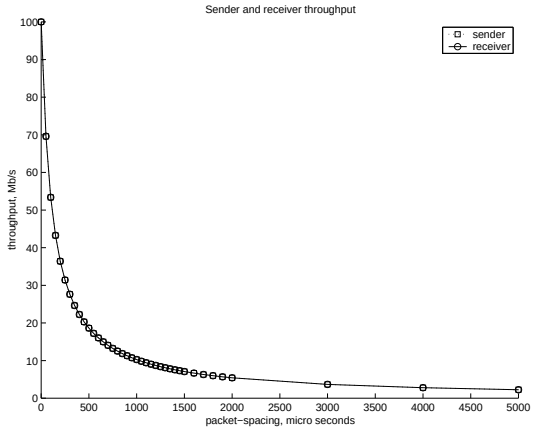

Figure 7. Throughput for One PSA

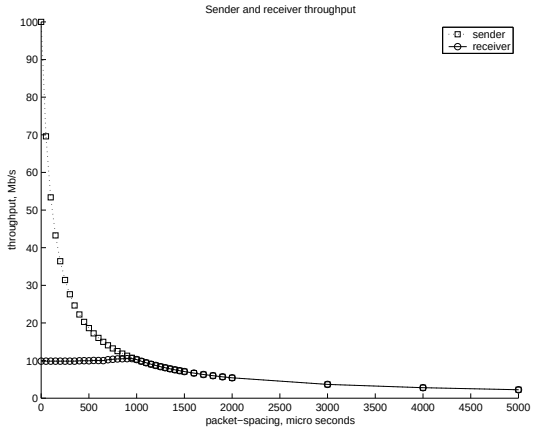

Figure 8. Throughput for One of the 16 PSAs

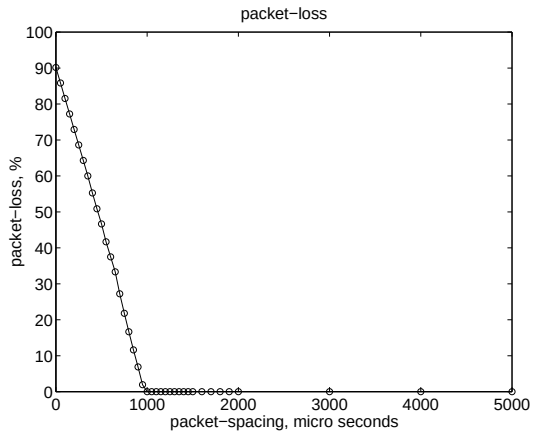

Figure 9. Packet Loss for One of 16 PSAs

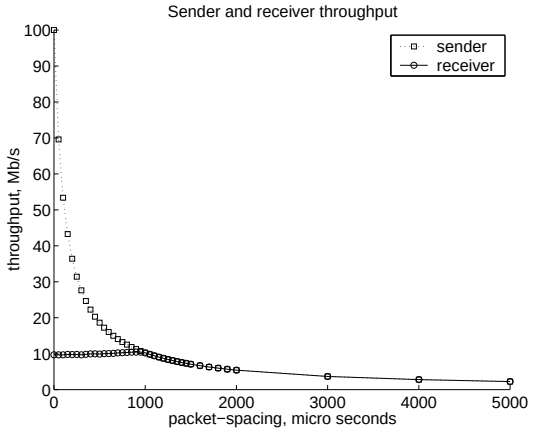

Figure 10. Throughput for One PSA of 16 PSAs and 16 TCP Connections

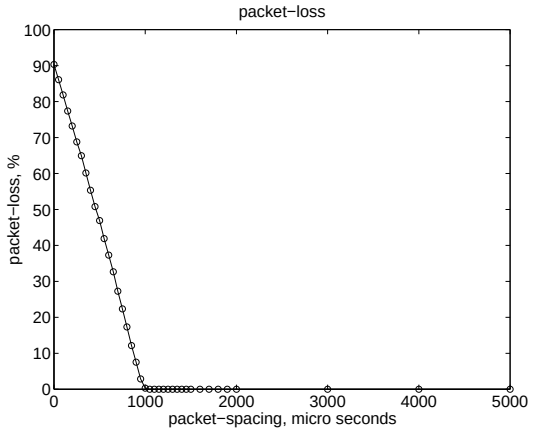

Figure 11. Packet Loss for One PSA of 16 PSAs and 16 TCP Connections

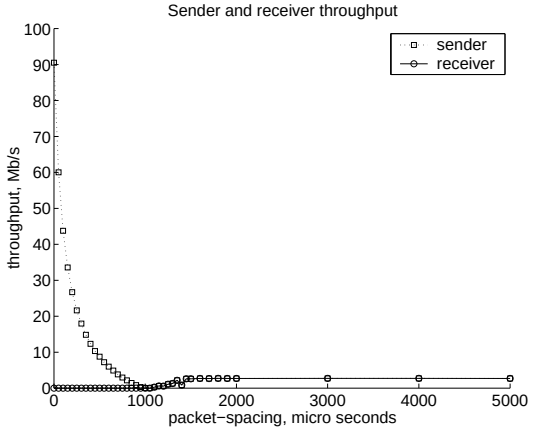

Figure 12. Throughput for One TCP of 16 TCP Connections and 16 PSAs

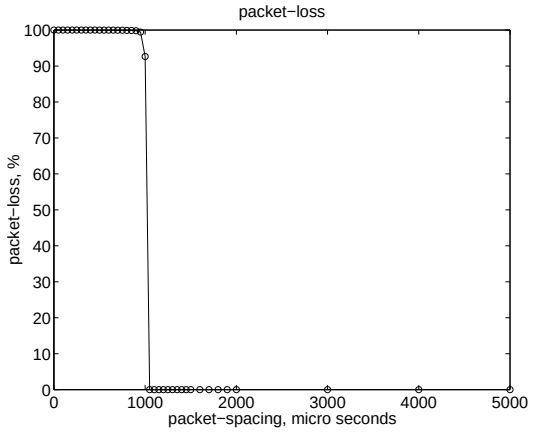

Figure 13. Packet Loss for One TCP of 16 TCP Connections and 16 PSAs

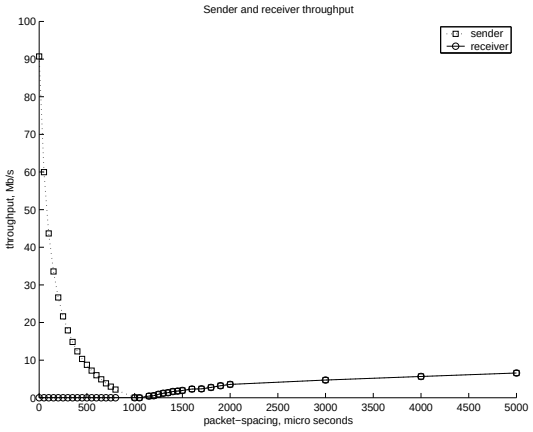

Figure 14. Throughput for One TCP (Window Size = 800 Packets) of 16 TCP Connections and 16 PSAs

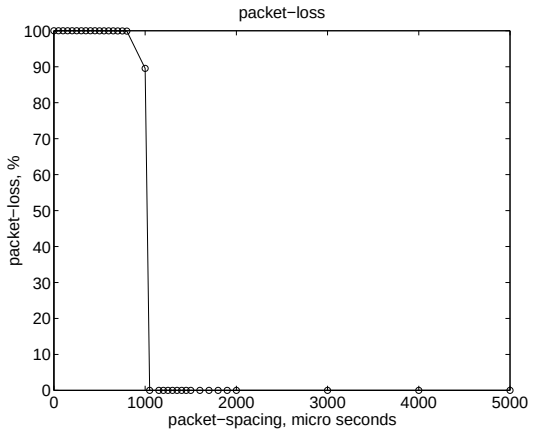

Figure 15. Packet Loss for One TCP (Window Size = 800 Packets) of 16 TCP Connections and 16 PSAs

time of packets at one PSA receiver, using the same experimental set-up as described in Section 3.2.3. The sending PSAs used a spacing of 1500 μs; the resulting inter-packet spacings at the receiver averaged 1540.6 μs with a standard deviation of 64.75 μs.

3.3 Adaptive PSA Simulations

Our adaptive PSAs implement the damped PSP, which tries to find the ideal packet spacing under varying network conditions. We first show the behavior of two adaptive PSAs competing against each other and then with two additional TCP connections. As in Section 3.2, the TCP connections were that of infinite file transfers.

3.3.1 Competing Adaptive PSAs

Figure 16 shows how the sending rate of an adaptive PSA varies with time. The adaptive PSA makes small oscillations around the ideal sending rate. Figure 17 demonstrates that our adaptive PSAs are fair (when both are started simultaneously) as both adaptive PSAs have sending rates that lie on the fairness line.

Figure 18 shows us a portion of the fairness graph where one adaptive PSA started 10 seconds later than the other. As we can see, both adaptive PSAs change their rates in a fair manner and eventually make small oscillations about the ideal sending rate.

3.3.2 Competing Adaptive PSAs with Background Traffic

In this simulation, we ran 10 TCP connections with infinite file transfers in the background and two adaptive PSAs competing in the foreground. Figure 19 shows that the adaptive PSAs respond readily to congestion. And again, both adaptive PSAs have very similar sending rates.

4 Conclusion

Perhaps the most interesting result in this paper is that a receiver's realizable throughput actually *increases* (up to a point) even when the sender's transmission rate decreases. This result has dramatic implications on many of today's multimedia applications that blast packets onto the network as fast as possible, i.e., no packet spacing. By slowing down the introduction of packets into the network, congestion is alleviated at the intermediate routers; this, in turn, results in a net increase in throughput. Thus, this work provides an incentive for multimedia provides not to blast UDP packets indiscriminately into the network. In addition, it provides motivation for the deployment of a packet-spaced protocol

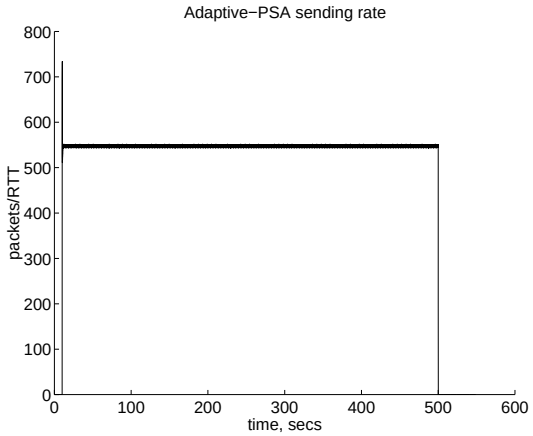

Figure 16. Sending Rate for One of Two Adaptive PSAs

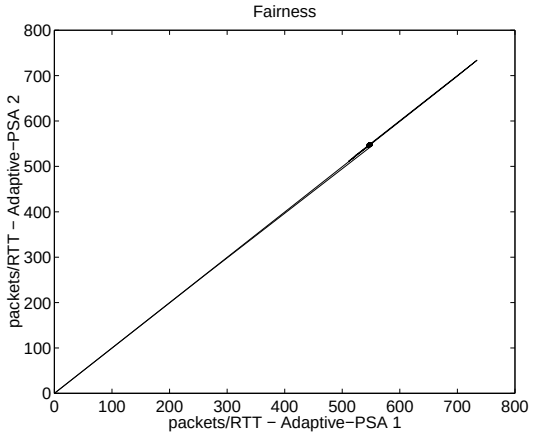

Figure 17. Fairness - Simultaneous Start

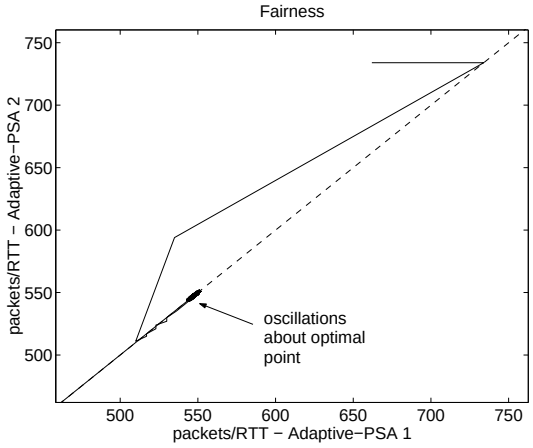

Figure 18. Fairness - Delayed Start

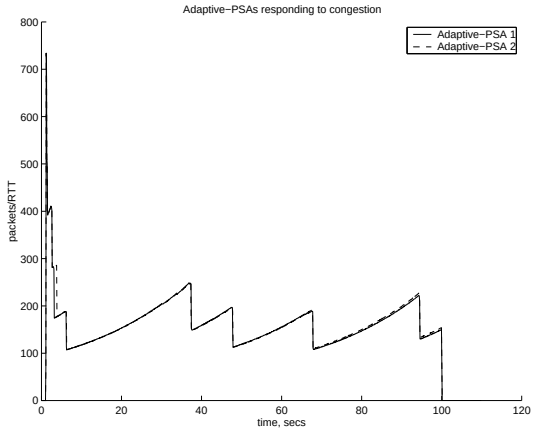

Figure 19. Two Adaptive PSAs Competing with 10 TCPs

that can deliver UDP-like performance yet still be responsive to competing connections.

Our damped packet-spacing protocol (PSP), implemented via an adaptive PSA, sends data near its "optimal" sending rate by using a simple feedback mechanism that reports packet loss every RTT. This mechanism in turn controls the amount of packet spacing. Our preliminary results demonstrate that by introducing packet spacing to a multimedia stream, packet loss can be reduced dramatically without much loss in throughput.

Future work includes examining the performance of our damped PSP with different types of application traffic. Of particular interest are those applications that generate data in short bursts with relatively large intervals between bursts. Based on the experimental results presented here, we expect that the packet loss that would normally be induced by these bursts to be greatly reduced.

References

[1] Ucb/lbnl/vint network simulator - ns (version 2). http://www.isi.edu/nsnam/vint/index.html.

[2] A. Aggarwal, S. Savage, and T. Anderson. Understanding the Performance of TCP Pacing. In *Proceedings of IEEE INFOCOM*, 2000.

[3] R. Braden. Recommendations on Queue Management and Congestion Avoidance in the Internet. Technical Report draft-irtf-e2e-queue-mgt00.txt, Internet Draft, March 1997.

[4] A. Feng. RAPID: Rate-Adjusting Protocol for Internet Delivery. Thesis Proposal (in progress), University of Illinois at Urbana-Champaign, 2001.

[5] W. Feng and W. Feng. The Impact of Active Queue Management on Multimedia Congestion Control. In *Proceedings of IC3N*, October 1998.

[6] W. Feng, D. Kandlur, D. Saha, and K. Shin. BLUE: A New Class of Active Queue Management Algorithms. Technical Report CSE-TR-387-99, University of Michigan, April 1999.

[7] W. Feng and P. Tinnakornsrisuphap. The Adverse Impact of the TCP Congestion-Control Mechanism in Heterogeneous Computing Systems. In *Proceedings of ICPP'00*, August 2000.

[8] W. Feng and P. Tinnakornsrisuphap. The Failure of TCP in Distributed Computational Grids. In *Proceedings of SC'00*, November 2000.

[9] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-Based Congestion Control for Unicast Applications. In *Proc. of SIGCOMM 2000*, August 2000.

[10] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.

[11] R. Jain. Myths about Congestion Management in High Speed Networks. *Internetworking: Research and Experience*, 29(2):101–113, February 1992.

[12] J. Mahdavi and S. Floyd. TCP-Friendly Unicast Rate-Based Flow Control. Technical report, Note sent to end2end-interest mailing list, January 1997.

[13] Microsoft. NetShow Theater Server White Paper. Available at *http://www.microsoft.com/Theater/overview.htm*, 1998.

[14] J. Mo, J. Walrand, and V. Anantharam. Analysis and Comparison of TCP Reno and Vegas. In *Proceedings of IEEE INFOCOM*, 1999.

[15] J. Padhye, J. Kurose, D. Towsley, and R. Koodli. A Model Based TCP-Friendly Rate Control Protocol. In *Proceedings of NOSSDAV'99*, 1999.

[16] A. Rangarajan and A. Acharya. ERUF: Early Regulation of Unresponsive Best-Effort Traffi c. In *Proceedings of ICNP'99*, October 1999.

[17] RealNetworks Inc. RBN^{TM} (Real Broadcast Network) White Paper. Available at *http://www.real.com/solutions/-rbn/whitepaper.html*, January 1999.

[18] RealNetworks Inc. RealVideo Technical White Paper. Available at *http://www.real.com/devzone/library/whitepapers/-overview.html*, January 1999.

[19] R. Rejaie, D. Estrin, and M. Handley. Quality Adaptation for Congestion Controlled Video Playback over the Internet. In *Proceedings of ACM SIGCOMM '99*, September 1999.

[20] R. Rejaie, M. Handley, and D. Estrin. An End-to-End Rate-Based Congestion Control Mechanism for Real-Time Streams in the Internet. In *Proceedings of INFOCOM'99*, March 1999.

[21] I. Rhee, V. Ozdemir, and Y. Yi. TEAR: TCP Emulation at Receivers — Flow Control for Multimedia Streaming. Technical report, North Carolina State University, April 2000.

[22] D. Sisalem and H. Schulzrinne. The Loss-Delay Based Adjustment Algorithm: A TCP-Friendly Adaptation Scheme. In *Proceedings of NOSSDAV'98*, 1998.

[23] D. Tan and A. Zakhor. Real-Time Internet Video Using Error Resilient Scalable Compression and TCP-Friendly Transport Protocol. *IEEE Transactions on Multimedia*, May 1999.

[24] P. Tinnakornsrisuphap, W. Feng, and I. Philp. On the Burstiness of the TCP Congestion-Control Mechanism in a Distributed Computing System. In *Proceedings of ICDCS'00*, April 2000.