The Impact of Active Queue Management on Multimedia Congestion Control

Wu-chang Feng
University of Michigan
Dept. of EECS
Ann Arbor, MI 48109
wuchang@eecs.umich.edu

Wu-chi Feng
Ohio State University
Dept. of Comp. and Info. Sci.
Columbus, OH 43210
wuchi@cis.ohio-state.edu

Abstract

Multimedia applications that transmit long continuous streams of data across the Internet will continue to stress networking technologies for the years to come. To help improve the performance of congestion avoidance protocols like TCP and to punish non-adaptive UDP-based applications, the IETF is considering the widespread deployment of Random Early Detection (RED) queue management. In this paper, we examine the impact of RED queue management on multimedia applications and their congestion control mechanisms. Our results show that even in ideal scenarios, the use of TCP-based congestion control in conjunction with active queue management based on RED will add significantly more bandwidth jitter over both short and long time scales, making it particularly difficult for adaptive, TCP-compatible, multimedia applications to converge on a single, fixed sending rate.

1. Introduction

The Internet has recently seen a substantial increase in the amount of multimedia traffic it carries due to the deployment of such applications as web conferencing and streaming audio/video. While some of these applications have built-in rate control, there have been increasingly many multimedia applications which do not. Since its deployment, TCP and its congestion control algorithms have been instrumental in preventing congestion collapse and in keeping packet loss rates low over the Internet. However, because of the proliferation an increasingly large number of non-adaptive multimedia applications using UDP, the Internet has recently seen a steady rise in loss rates across various network links [8]. To combat rising packet loss rates, the Internet Engineering Task Force (IETF) is considering widespread deployment of active queue management techniques to improve the performance of congestion controlled applications and to punish non-adaptive applications. The techniques being considered are based on using Random Early Detection (RED) [2,4] along with additional mechanisms to identify malicious flows and force them to maintain a fair share of the available bandwidth.

The widespread deployment of active queue management mechanisms will have important implications for multimedia congestion control. One clear impact is that such applications will now be forced to develop rate control mechanisms that are compatible with TCP in order to avoid being penalized by active queue management. Applications which are less aggressive than TCP will get less than their fair share of available bandwidth while applications which are more aggressive than TCP will see their performance degrade as active queue management identifies and penalizes them accordingly. This paper examines another, more subtle, impact that active queue management schemes will have on end-host applications and congestion control. In particular, we show that even in ideal scenarios, the use of TCP-based congestion control in conjunction with active queue management based on RED will add significantly more bandwidth jitter over both short and long time scales, making it particularly difficult for adaptive, TCP-compatible, multimedia applications to converge on a single, fixed sending rate. Without modifying the rate and congestion control mechanisms used, this can significantly impact the performance of multimedia applications.

0-8186-9014-3/98 $10.00 © 1998 IEEE

In the following section, we review some of the relevant TCP background including a discussion of Random Early Detection queue management. In Section 3, we describe the experiments that we conducted with RED queue and drop-tail queue management within the network. In addition, we describe its impact on video-based applications. Finally, we summarize our findings and provide directions for future work.

2. TCP

Over the last decade, TCP congestion control has been used to effectively regulate the rates of individual connections sharing network links. TCP congestion control is window-based. The sender keeps a congestion window whose size limits the number of unacknowledged packets it can have outstanding in the network. Upon receiving acknowledgments for successfully transmitted data, the sender increases its transmission rate by incrementing the size of its congestion window. At some point in time, the transmission rate eventually exceeds the network's capacity. When this happens, queues build up in the routers and overflow, causing packets to be dropped. TCP assumes that packet loss is due to congestion and reduces its congestion window upon detecting the loss. One way in which TCP detects a packet loss is through the receipt of a number of duplicate acknowledgments from the receiver[5]. Upon receiving a given number of duplicate acknowledgments, TCP infers that a packet loss has occurred and immediately halves its transmission rate by halving its congestion window. When congestion is severe enough such that packet loss cannot be inferred in such a manner, TCP relies on a retransmission timeout mechanism to trigger subsequent retransmissions of lost packets. When a retransmission timer is triggered, TCP reduces its window size to one segment and retransmits the lost segment.

One problem with the TCP congestion control algorithm over current, drop-tail networks is that the TCP sources back off their rates only after detecting packet loss due to queue overflow. This is a problem since considerable time may pass between the packet drop at the router and its detection at the source. In the meantime, a large number of packets may be dropped as the senders continue to transmit at a rate that the network cannot support. Active queue management has been proposed as a solution for preventing losses due to buffer overflow. The idea behind active queue management is to detect incipient congestion *early* and convey congestion notification to the end-hosts, allowing them to back off before queue overflow and sustained packet loss occur. One form of active queue management being proposed by the IETF for deployment in the network is RED [2,4]. RED maintains a weighted average of the queue length which it uses to detect congestion. When the average queue length exceeds a minimum threshold (min_{th}), packets are randomly dropped with a given probability. The probability that a packet arriving at the RED queue is dropped depends on, among other things, the average queue length, the time elapsed since the last packet was dropped, and an initial probability parameter (max_p). When the average queue length exceeds a maximum threshold (max_{th}), all packets are dropped.

3. Performance Evaluation

In this section we describe the experiments that we have conducted. The first set of experiments describes the bandwidth that is achievable through drop-tail and RED managed queues. The second set of experiments describes the impact of drop-tail and RED on multimedia applications

3.1. Drop-tail versus Random Early Detection queue management

There are distinct differences in the behavior of TCP sources depending on the queue management scheme being used. With drop-tail queues, TCP sources continually increase their sending rates until their sending rates exceed the capacity of the bottleneck link. When this occurs, the buffers at the bottleneck link overflow and packets are dropped for a period of time across all connections. Upon detecting packet loss, each connection cuts back its sending rate causing the offered load to drop below the link's capacity and allowing the bottleneck queue to drain. With RED queues, the dynamics of congestion control are different. In this case, TCP sources still continually increase their sending rates, however, instead of waiting for the buffers at the bottleneck to fill up before dropping packets, the RED queue detects an increase in queue length, randomly selects a packet from a particular connection, and drops it. Packet drops are spaced out in time causing one connection to back off at a time while the other connections continue. While the randomness of the RED algorithm prevents problems with synchroniza-

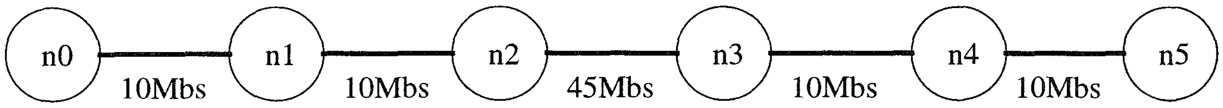
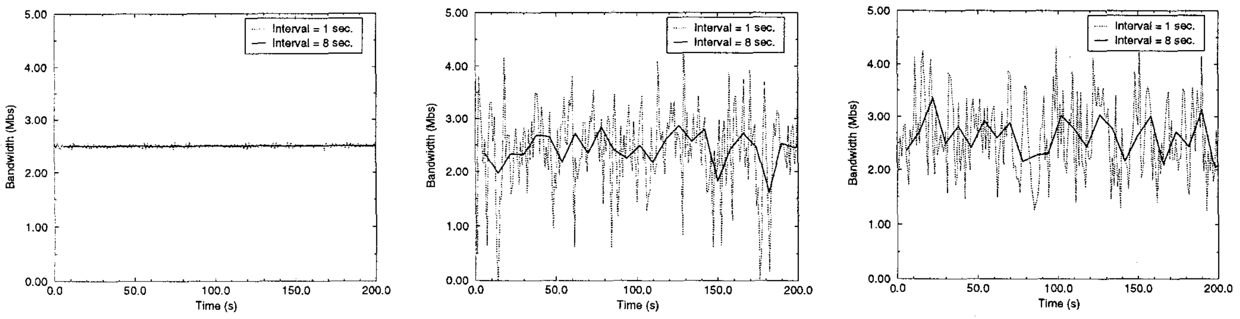


Figure 1: Network Topology



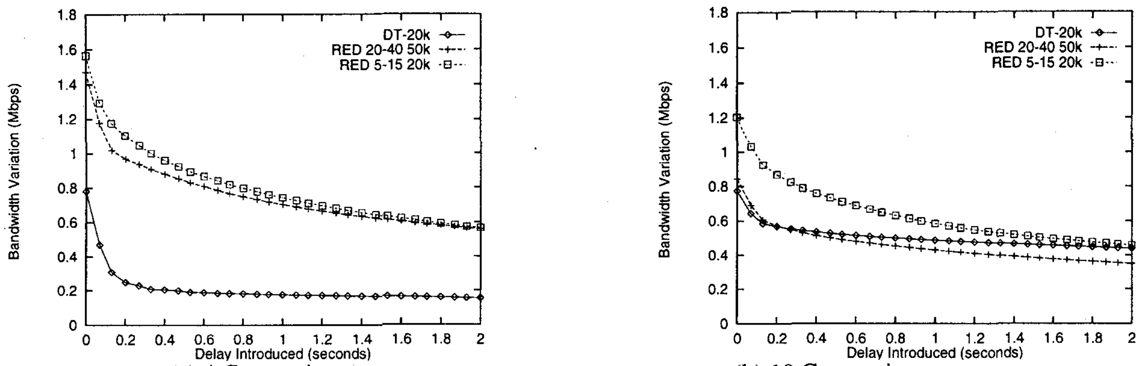
(a) Drop-tail (20 KB) (b) RED (20 KB) (c) RED (50 KB)

Figure 2: Bandwidth Plots for 4 Connections

tion[3], it can also cause significant bandwidth jitter to individual connections.

In order to examine the impact of the RED algorithm on the bandwidth obtained using TCP-based congestion control, we evaluated the performance of several TCP sources in an idealized network setting using the *ns* simulator[6]. Figure 1 shows the network configuration used. Each link is labeled with its capacity and has a transmission delay of *1 ms*. Given this network, we then run a number of TCP connections from node *n0* to node *n5* and examine the bandwidth delivered to each connection over varying periods of time. In order to more fairly compare the performance between drop-tail and RED queues we ran each experiment under three different router configurations where the routers are equipped with: drop-tail queues of *20KB*; RED queues of *20KB* (min_{th} = *5KB*, max_{th} = *15KB*); and RED queues of *50KB* (min_{th} = *20KB*, max_{th} = *40KB*). We varied the RED queue because the RED algorithm tends to minimize queue lengths at congested routers. While this is good for minimizing delay, it also has the effect of reducing the overall bandwidth-delay product of the network. As a result, the window sizes of individual TCP connections going across the network are smaller and thus, more susceptible to retransmission timeouts[7]. Retransmission timeouts have been shown to severely degrade the fairness of bandwidth sharing between multiple TCP connections. To eliminate the effects that retransmission timeouts have and to isolate the effects of the RED algorithm, we evaluate the performance using a RED queue whose min_{th} is equal to the size of the drop-tail queue and whose capacity is more than twice the size.

We first examine the performance using a small number of connections. Figure 2 shows the results with four active connections. Note that in this configuration, because of the small number of sources going across the network, each TCP source keeps a congestion window which is sufficiently large to maintain ACK-clocking and avoid retransmission timeouts. For heavier workloads, the amount of network buffers at the bottleneck link is on the order of the number of connections going through it. As a result, the individual congestion window sizes of each TCP connection is small and as described above, any packet loss induces retransmission timeouts which causes significant bandwidth jitter to TCP sources. Figure 2(a) shows the bandwidth received by a single TCP source over 1 and 8 second time intervals using drop-tail queues in the network. Figure 2(b) and Figure 2(c) show the bandwidth received by a single TCP source over the same intervals using *20KB* and the *50KB* RED queues respectively. As the figures show, even in an ideal setting,



(a) 4 Connections (b) 10 Connections

Figure 3: Reducing the Bandwidth Variation

RED queue management causes considerable bandwidth jitter when used in conjunction with TCP congestion control. The bandwidth received by the source fluctuates considerably around its fair share of the link capacity (*2.5 Mbs*). In contrast, the performance over drop-tail queues remains quite predictable, a result which is consistent with recent observations made on a fairly loaded web server[1].

3.2. Queue management and multimedia applications

Video based applications which are adaptive to network conditions will be impacted by RED queue management. In particular, these video applications typically require a *target* bandwidth for which to encode the video stream. This target is then used by the application to either reduce the frame rate or frame quality delivered across the network. Under RED queue management, large end-to-end bandwidth variations make it more difficult for video *applications* to target a specific bandwidth in which to encode the video. In order to reduce the bandwidth variation for the end applications, we can *smooth* the bandwidth requirements over time by using additional buffers and delaying playback at the receiver. That is, the TCP connection is buffered enough to allow the target bandwidth to be stable for a reasonable amount of time. The main drawback, however, is that this buffering introduces delay between the transmission and playback of the video. This delay can make it hard to provide interactive applications like video conferencing.

Figure 3 shows the amount of delay that would be required to reduce the variation under both drop-tail and RED queues. Figure 3(a) shows the bandwidth variation reduction for 4 connections. Assuming no smoothing of bandwidth requirements, the figure shows that the bandwidth variation seen by the application when drop-tail queues are used is about *0.8 Mbs*. In order for the application to see the same bandwidth variation over RED queues, it would have to delay the video approximately *0.9* seconds. That is, the *horizontal distance* between the drop-tail and RED graphs represents the amount of extra delay required to be build into the application to achieve similar bandwidth targets. Assuming the TCP source running over the drop-tail queues does even a minimal amount of smoothing (say of 2 frames), the RED queues would require a significant amount of buffering delay to remove the bandwidth variation. Figure 3(b) shows the bandwidth variation using a larger number of connections. In this experiment, the number of connections is large enough so that retransmission timeouts occur when packets are dropped for both the drop-tail and the RED queues. As the figure shows, retransmission timeouts cause significant bandwidth variation among sources regardless of the queue management scheme.

4. Acknowledgements

This work was supported in part by the Ohio State University Research Foundation and the Ameritech Corporation. The results presented in this paper do not necessarily reflect that of the funding agencies.

5. Conclusion

This paper has shown that the use of active queue management schemes based on RED can cause significant bandwidth variation to TCP-compatible congestion control. This bandwidth variation can significantly degrade the performance of congestion-controlled applications. For multimedia applications, RED queue management will require significant buffering (and delay) in order to provide the application with a reasonably stable bandwidth target. As part of ongoing work, we are currently examining smoother rate/congestion control algorithms which alleviate this problem while maintaining TCP compatibility over longer time intervals.

6. References

[1] H. Balakrishnan, S. Seshan, M. Stemm, and R. Katz. Analyzing Stability in Wide-Area Network Performance. In Proc. of SIGMETRICS, June 1997.

[2] R. Braden, D. Clark, et. al. Recommendations on Queue Management and Congestion Avoidance in the Internet. Internet Draft draft-irtf-e2e-queue-mgt-00.txt, March 1997.

[3] S. Floyd and V. Jacobson. On Traffic Phase Effects in Packet-Switched Gateways. Internetworking: Research and Experience, 3(3):115--156, September 1992.

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

[5] V. Jacobson. Modified TCP Congestion Avoidance Algorithm. end2end-interest mailing list (ftp://ftp.ee.lbl.gov/email/vanj.90apr30.txt), April 1990.

[6] S. McCanne and S. Floyd. http://www-nrg.ee.lbl.gov/ns/. ns-LBNL Network Simulator, 1996.

[7] R. Morris. TCP Behavior with Many Flows. In Proc. IEEE International Conference on Network Protocols, October 1997.

[8] V. Paxson. Automated Packet Trace Analysis of TCP Implementations. In Proc. of ACM SIGCOMM, September 1997.