A Better-Than-Best-Effort Service for Continuous Media UDP Flows*

Mark Parris Kevin Jeffay F. Donelson Smith Jan Borgersen

University of North Carolina at Chapel Hill
Department of Computer Science
Chapel Hill, NC 27599-3175 USA
{parris,jeffay,smithfd,borg}@cs.unc.edu

http://www.cs.unc.edu/Research/dirt

Abstract

The Internet community is very interested in addressing congestion with active queue management mechanisms like Random Early Detection (RED). Recent proposals extend these mechanisms to actively penalize "misbehaving" flows. These methods favor TCP and TCP-like flows strongly over other flows like UDP. In this paper we propose extensions to active queue management disciplines which continue to address congestion and reward TCP-friendly flows while minimizing impact on continuous media flows using other protocols. Our mechanism, Drop Preference Management (DPM), recognizes tagged flows and manages their latency while constraining the bandwidth they consume. We present empirical results of experiments comparing our mechanism to plain RED.

1. Overview

As the Internet continues to evolve, increasing attention is being given to recognizing and addressing congestion within the network. Particularly, there is increasing focus on recognizing "well-behaved" flows, those that respond to congestion by reducing the load they place on the network. Both Braden *et al*., and Floyd *et al*., recognize TCP flows with correct congestion avoidance implementations as well behaved [2, 6, 7, 1]. They argue that these flows, as "good network citizens," should be protected and isolated from the effects of "misbehaving" flows. Misbehaving flows include incorrect implementations of TCP, unresponsive but lightweight UDP connections, and high bandwidth flows that aggressively retransmit when drops occur. A recent internet draft considers the problem of congestion in the current internet and makes two recommendations [2]. First, they recommend deploying active queue management schemes, specifically Random Early Detection (RED) to more effectively notify responsive flows of congestion [5]. Active queue management refers to extending the queueing discipline in the router beyond simple enqueue and dequeue with drop-tail when full. For example, RED does not wait until the queue is full to drop packets. Instead, it probabilistically drops incoming packets when the queue's average size exceeds a particular threshold and automatically drops a random packet when the average exceeds a higher threshold. This provides earlier feedback, before the queue overflows, and causes higher bandwidth flows to see a greater number of drops. Second, they recommend continued development of mechanisms to deal with flows that do not respond to congestion in TCP-friendly manner. To date "dealing with" these other flows has centered on how to constrain or penalize those flows [6, 9]. We recognize the need to protect well-behaved flows but also recognize that many applications choose unresponsive protocols, such as UDP, because they are concerned with throughput and (especially) latency rather than reliable delivery. Since reliable delivery in TCP depends on timeouts, feedback, and retransmissions, it can be incompatible with performance goals. Multimedia applications are a prime example of applications that avoid TCP for performance reasons.

Simply penalizing these non-TCP flows leaves application developers with some unattractive options. With the deployment of RED in many routers, application developers must realize UDP flows will be subject to more aggressive drop policies than in the past. The developer could use TCP and incur overhead for features she may not want. Or, she could use another protocol and be subject to aggressive drop policies. Another alternative would be to use a protocol that implements TCP-like congestion management without the other undesired features such as reliable

* Supported by grants from the National Science Foundation (grants IRIS-9508514 & CCR-9510156), the Advanced Research Projects Agency (grant number 96-06580), the IBM Corporation, and a graduate fellowship from the Intel Corporation.

delivery [3]. The correct long-term solution is to develop adaptive applications that respond to congestion notification (as well as application level feedback). Our approach recognizes all of these solutions. Our solution is orthogonal, providing better support for continuous media flows in the router.

To mitigate the impact of active queue management on UDP flows, we are working on queue management policies for routers that attempt to balance the concerns of congestion avoidance and the requirements for continuous media applications using UDP. Specifically, we are experimenting with extensions to the Random Early Detection (RED) packet discard mechanism for providing better performance for UDP flows without sacrificing performance for TCP flows. The following briefly outlines the design of our RED extension, called RED with Drop Preference Management (RED-DPM), and presents some early empirical results that suggest the mechanism performs effectively.

2. Active Queue Management

RED was designed as a mechanism to notify responsive flows of congestion, either explicitly, by marking packets, or implicitly, by dropping packets of a flow. The likelihood of a flow being notified is directly related to its (average) bandwidth utilization. In this way RED avoids global synchronization as many flows back off at the same time, and avoids unfairly penalizing bursty traffic. Instead, those flows consuming the most bandwidth see the most drops. If responsive, the flow will reduce the load it generates and the number of drops on that flow will decrease, allowing it to reach an equilibrium state. If the flow is unresponsive, the flow will continue to see a high number of drops, corresponding to its utilization of a large part of the queue. However, RED assumes that traffic is responsive and thus is vulnerable to misbehaving unresponsive flows. A non-responsive flow could consume a large share of the bandwidth while other flows decrease their utilization in response to congestion notification. While RED does drop packets in ratio to the flow's arrival rate, it still drops an equal percentage of arriving packets from all flows. So even well behaved flows continue to see drops if one misbehaving flow keeps the queue full. As a result of these drops, well-behaved flows can back down to extremely low (or no) throughput. Both RED with Penalty Boxes and FRED address this deficiency in RED [6, 9]. Floyd and Fall propose mechanisms for identifying several classes of flows: TCP-friendly, unresponsive, and very-high bandwidth flows [6]. They suggest simply that those flows which are not TCP-friendly should be "regulated" (without clearly defining a mechanism for regulation) to prevent them from dominating network resources while TCP-friendly (*i.e.*, responsive) flows continue to be notified of congestion via a RED-like mechanism.

FRED is an extension to RED that uses per-active-connection accounting to ensure fairness and isolation between all active flows [9]. While RED drops an equal percentage of packets across all flows, FRED increases the likelihood of drops for high bandwidth flows. It does this by concentrating drop actions on those flows that are exceeding a threshold based on the average *per flow* logical queue length. Further, flows whose logical queue utilization is below average will not be subject to these drops. It also concentrates drop actions on those flows that have failed to respond to notification in the past. Both FRED and RED with penalty boxes take a harsh stance on unresponsive flows, constraining them tightly by dropping potentially all arriving packets once a queue in a router is above threshold.

3. RED with Drop Preference Management

We propose a mechanism to constrain misbehaving flows while attempting to provide them with the best performance possible under those constraints. A key observation is that latency may be as important a consideration for these flows as overall bandwidth. Thus we trade-off bandwidth for (lower) latency. We seek to minimize the latency of those packets that do arrive at the end system and avoid delivery of *stale* packets, packets that are queued in the network and are not likely to arrive in time to be useful.

In our scheme we identify multimedia flows with a tag[1] and maintain per flow statistics for those flows only. Non-tagged flows are subject to the standard RED queue management policy. Packets in tagged flows are subject to drops based on queue length thresholds similar to FRED but also are subject to drops if packets for that queue exceed a "staleness" threshold. The staleness threshold is based on the age and depth of the oldest packet for this flow currently enqueued on the router. Further, anytime a decision to drop a packet is made, an alternative drop mechanism, *delete and advance*, is used. This mechanism drops the oldest packet (at the head of the flow's logical queue) so newer, *fresher* packets are given priority. It also reduces the time the new packets spend in the queue by

[1] Any of several proposed tagging mechanisms in the IP header could be used, *e.g.*, the Clark or Nichols schemes [4, 10].

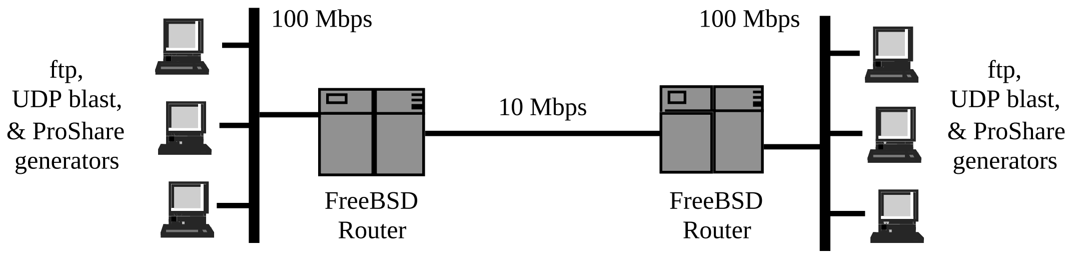


Figure 1: Experimental network configuration.

promoting them into their logical predecessor's location in the main queue. The depth of the queue elements for all other flows is unchanged.

With drop preference management we expect to reduce the impact of misbehaving tagged flows on responsive flows by performing more aggressive dropping of non-TCP-conformant flows. These flows are subjected to a RED test on their (potentially much shorter) logical queue as well as a staleness test. However, UDP flows, and in particular responsive UDP flows, benefit overall as their packets tend to stay closer to the head of the router queue and thus are delivered with lower latency. DPM should benefit both responsive and non-responsive continuous media flows. Non-responsive flows will see a large number of drops, just as they would in the face of most congestion, but the packets that are delivered will have lower latency (because of the delete and advance drop policy). Responsive flows will see this benefit but they will also be notified sooner of congestion because the oldest, not the newest, enqueued packet will be the one dropped. And most importantly, a responsive flow will see no drops once it has adjusted its load so that it consumes a fair share of the queue. Low bandwidth non-responsive flows will see this benefit as well.

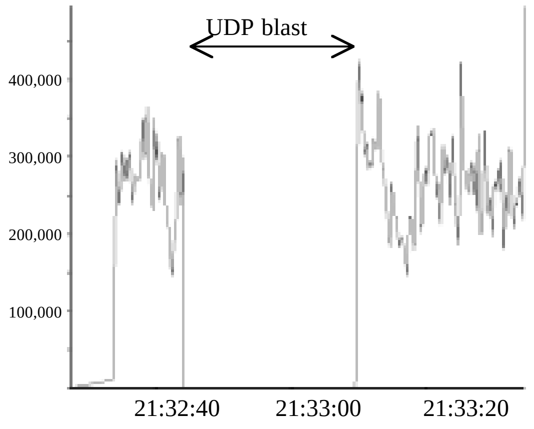
4. Empirical Results

We have implemented RED-DPM within the ALTQ version of the FreeBSD [8]. The RED-DPM implementation maintains a logical queue for each tagged flow and performs a threshold and staleness test on each tagged flow's logical queue immediately prior to the normal RED test.

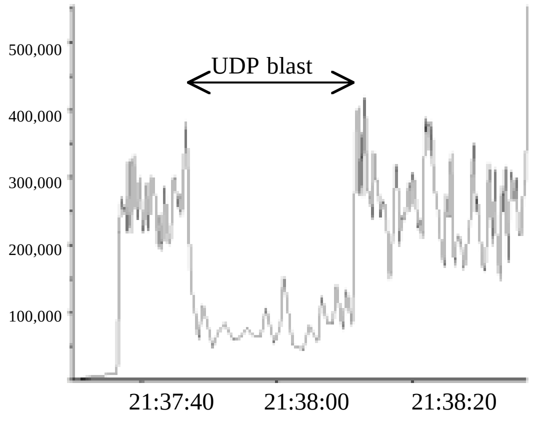
To test the implementation we have constructed a simple network consisting of a two switched 100 Mbps Ethernet LANs that are interconnected by a 10 Mbps Ethernet. FreeBSD routers route traffic between the 100 Mbps Ethernets across the 10 Mbps Ethernet as shown in Figure 1. This 10Mbs network creates a bottleneck to serve as a point of congestion. A series of machines at the edges of the network establish a number of connections to machines on the opposite side of the network. Connections include a mix of TCP connections and UDP connections. In particular, several of the UDP connections are videoconferencing flows generated by a version of the Intel ProShare system (6 flows at approximately 200Kbps each). We also generate a "UDP blast" of unresponsive UDP traffic sent from one machine at the maximum rate available.

Figure 2 reports some of our early results. It compares the performance of TCP (3 machines sending bulk data at the maximum rate available) and UDP connections (Proshare) when the routers use RED queue management and RED-DPM management. Figures 2a and 2b, show the throughput of the TCP connections, first with just TCP and Proshare, then during a period when an unresponsive UDP blast occurs, and finally back to just the TCP and Proshare. Figure 2a shows the performance under RED queue management. Figure 2b shows the performance under RED-DPM queue management. As mentioned above, RED does not effectively protect responsive flows from non-responsive ones. During the UDP blast period, the TCP flows see so many drops that they back off entirely, getting no throughput. Figure 2a confirms this shortcoming. RED-DPM does provide protection by limiting the number of buffers that the unresponsive stream can occupy, thereby ensuring a minimal amount of space exists for TCP flows. The result, as illustrated in Figure 2b is improved throughput for TCP.

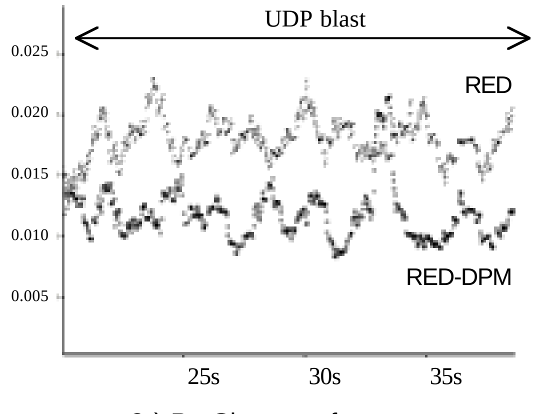
Figure 2c illustrates the impact of RED-DPM on the ProShare (UDP) flows. This shows the latency of the Proshare streams during the time the UDP blast is running. The top set of points is the latency with RED, while the lower, darker points are the latency with RED-DPM. In the case of RED-DPM, although the streams exhibit loss the delete-and-advance mechanism results in lower end-to-end latency than when plain RED is used.



2a) TCP performance under RED.
(throughput (bps) *v.* time)



2b) TCP performance under RED-DPM.
(throughput (bps) *v.* time)



2c) ProShare performance.
(end-to-end latency (secs) *v.* time)

Figure 2: Experimental results. Performance of TCP and UDP flows during a UDP "blast."

5. Conclusions and Future Work

We have shown preliminary results which indicate that current active queue management mechanisms, such as RED, can be extended to provide better support for continuous media flows while maintaining or exceeding the current performance offered to "well-behaved" flows. We intend to conduct further comparisons between DPM and FRED, conduct more experiments with different network configurations and traffic patterns, as well as exploring the design space available for thresholds and staleness values.

6. References

[1] R. Braden, Ed., *Requirements for Internet Hosts-Communication Layers,* RFC-1122, October 1989

[2] B. Braden, *et al., Recommendations on Queue Management and Congestion Avoidance in the Internet*, Internet draft draft-irtf-e2e-queue-mgt-01, Feb. 17, 1998.

[3] S. Cen, C. Pu, J. Walpole, *Flow and Congestion Control for Internet Streaming Applications*, Proc. SPIE/ACM Multimedia Computing and Networking '98, San Jose, CA, January 1998, pages 250-264.

[4] D. Clark, J. Wroclawski, *An Approach to Service Allocation in the Internet*, Internet Draft draft-clark-diff-svc-alloc-00, July 1997

[5] S. Floyd, & V. Jacobson, *Random Early Detection gateways for Congestion Avoidance*, IEEE/ACM Trans. on Networking, V.1 N.4, August 1993, p. 397-413.

[6] S. Floyd, S., & K. Fall, *Promoting the Use of End-to-End Congestion Control in the Internet*, February 1998. (Under submission to IEEE/ACM Trans. on Networking.)

[7] V. Jacobson, *Congestion Avoidance and Control*, ACM SIGCOMM '88, August 1988.

[8] C. Kenjiro, *A Framework for Alternate Queueing: Towards Traffic Management by PC-UNIX Based Routers*, Accepted to USENIX '98, Annual Technical Conference, New Orleans, LA, June 1998.

[9] D. Lin & R. Morris, *Dynamics of Random Early Detection*, Proc. SIGCOMM '97.

[10] K. Nichols, V. Jacobson , & L. Zhang, *A Two-bit Differentiated Services Architecture for the Internet*, Internet-Draft, draft-nichols-diff-svc-arch-00.txt, Nov, 1997.