Towards a Better-Than-Best-Effort Forwarding Service for Multimedia Flows

Kevin Jeffay
Department of Computer Science
University of North Carolina at Chapel Hill
Chapel Hill, NC 27599-3175
jeffay@cs.unc.edu
November 1999

1. Introduction & Motivation

Our research at UNC concerns the management of multimedia and more general continuous-media flows across the Internet. We focus on network support for (highly) interactive applications where real-time transmission is essential for the correct operation of the application. Such applications include tele-immersion applications and interactive distributed virtual environments. In these applications minimum possible end-to-end transmission delay is required for effective device control and for maintaining the sense of immersion in a virtual world [1, 2]. To realize the performance requirements of these and other real-time multimedia applications, end-system media adaptation techniques are commonly employed to ameliorate the effects of contention for bandwidth and other resources in the network. Most multimedia and virtual environment systems, including the ones we consider, are capable of adapting to modest changes in end-to-end network delay and throughput, however, doing so often reduces the function of the application.

Media adaptation can be abstractly described as the process of dynamically modifying an application's network bit- and packet-rate based on feedback from receivers [3]. Bit-rates can be modified by changing media generation parameters, media encodings, or levels of redundancy in the bit-stream [4]. Packet-rates can be manipulated by changing how the bit-stream is partitioned into network packets by changing the number of encoded media samples carried in a single packet [5]. The goal of all media adaptation schemes is to discover an operating point for the application that is sustainable given the current level of contention for network and end-system resources, and results in an acceptable mode of operation for the application.

Media adaptation is a form of application-level congestion control. Whereas other network applications such as Web browsers rely on the underlying network transport protocol, most notably TCP, for congestion control and avoidance, real-time interactive multimedia applications eschew TCP in favor of UDP. These applications either perform no congestion control or employ application-level techniques. The reliance on UDP by interactive multimedia applications typically results from a conscious trade-off between reliable, in-order data delivery and the ability to exercise control (however effective) over the reaction to packet loss and the variation in the delay of transmissions. This trade-off is made because a salient feature of real-time multimedia applications is the requirement that they transmit data continuously at a rate determined by the application. Real-time multimedia applications typically continuously sample the external environment. To maintain interactivity, continuous transmission of samples with low end-to-end delay is required.

The requirement of continuous transmission puts many multimedia applications at odds with current and proposed Internet network management practices and is at the core of our research. There exists a fundamental tension between the current practice of relying on end-systems to limit the rate at which they inject traffic into the network and the desire of many multimedia applications to transmit data continuously at a quasi-uniform rate. TCP connections detect packet loss in the network, treat it as a signal of congestion, and react by reducing their transmission rate. In the worst case, a connection will stop transmitting data for a time (on the order of 1-2 seconds) and then transmit packets at a minimal rate determined by the time it takes for a packet to transit the network and for an acknowledgement to return to the sender. While this particular reaction to packet loss is unsuitable for applications requiring continuous transmission at uniform rates, it is essential for the health of the Internet. It ensures quasi-fair sharing of link capacity between TCP connections and ensures that a pathological condition called *congestion collapse* does not occur. These are vital attributes given that TCP traffic accounts for approximately 97% of all the bytes transmitted over the Internet today [6].

UDP applications, for example high-bandwidth multimedia applications, have the potential to starve TCP connections of bandwidth. A network becomes congested when the flow of data into the network exceeds the capacity of the network to transmit data. Queues of packets build up in routers and switches and packets are dropped ("lost") when the queues overflow. A UDP application that does not respond to packet loss as an indicator of congestion will continue transmitting data into queues that are overflowing. TCP connections that share these queues will react to packet loss and reduce their transmissions. If the network resources previously consumed by these TCP connections (*e.g.*, buffer space in routers) are now consumed by the non-responsive applications, then the situation spirals. Router queues remain full and TCP connections, although transmitting at a lower rate, still lose packets and therefore continue to reduce their transmissions rate until they effectively stop transmitting.

Figure 1 illustrates this effect for a set of TCP connections that share a 10 Mbps link with a high-bandwidth UDP flow. The plot shows aggregate TCP throughput over time. Initially, a group of approximately 3,000 users browsing the Web share the 10 Mbps link. There are a sufficient number of browsers active at any one time to keep the average link utilization at approximately 95%. At time 30, some multimedia applications, a set of videoconferences in this case, commence execution. TCP throughput is slightly reduced by this overall increase in network traffic. At time 90, a single high-bandwidth UDP application commences transmission. Instead of sharing link capacity fairly with the TCP connections (*e.g.*, instead of consuming approximately $1/n^{\text{th}}$ of the link capacity where n is the number of active flows), the UDP application causes congestion and starves the TCP connections of bandwidth. The introduction of the unresponsive application causes the network to become saturated and hence causes packet loss. The TCP connections react to the loss by reducing their transmission rates and hence

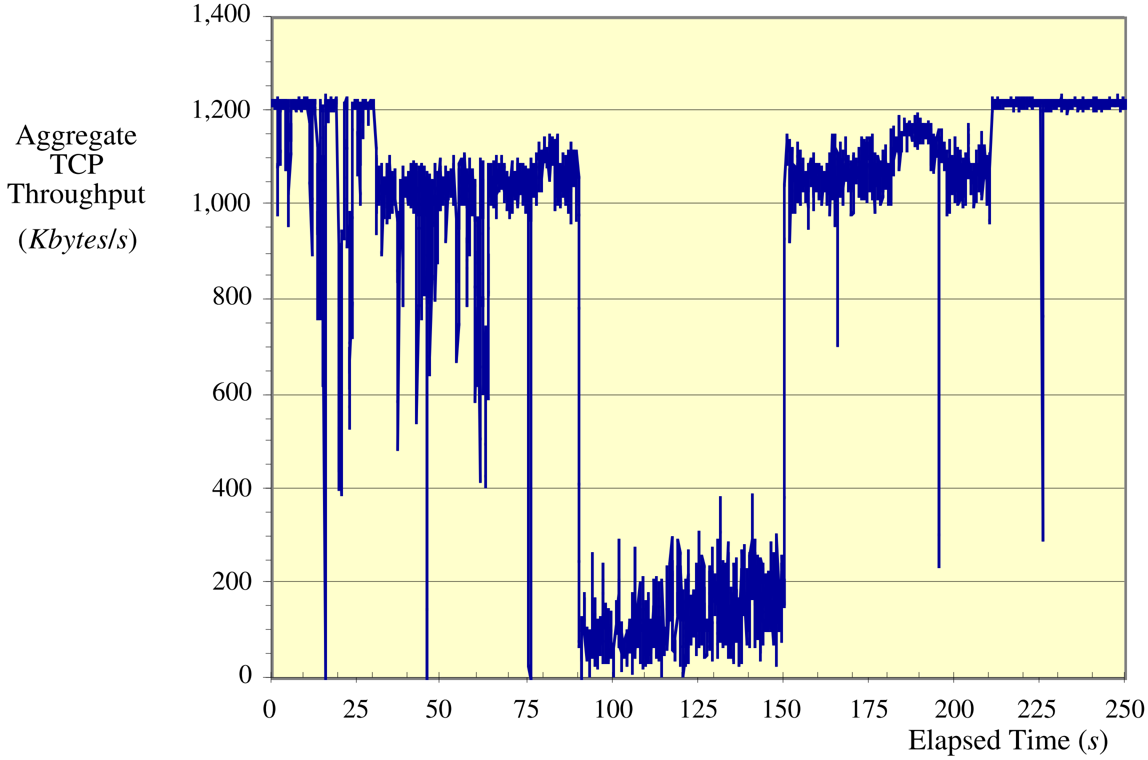

Figure 1: TCP starvation in the presence of unresponsive, high-bandwidth UDP flows.

consuming fewer buffers in the router. The UDP application also sees a high rate of packet loss but nonetheless does not adapt its transmissions. As a result, the buffers in the router remain full and all network connections continue to experience packet loss. The queue in the router remains full because in essence, the queue elements previously occupied by TCP packets are now occupied by UDP packets.

This example illustrates the negative effects of the interaction between the requirement for uniform-rate transmission and the need to be reactive to packet loss as an indicator of network congestion. Our research attempts to develop network resource management mechanisms that strike a balance between the conflicting requirements of continuous transmission and congestion control. TCP connections require isolation or protection from non-reactive applications (or more generally, from all non-TCP applications) and interactive UDP-based multimedia applications require the ability to transmit data continuously with (ideally) low end-to-end latency.

All of these goals can be met in a tunable manner by reserving bandwidth in the network for either individual connections or collections of connections, and explicitly allocating network bandwidth on a packet-by-packet basis by scheduling packets across network links. However, this approach is complex and requires close coordination among, and integration between, all routers (and hence all network service providers) along the path from sender to receiver. We are investigating an alternate approach based on queue management rather than packet scheduling. The practice of manipulating queues to bias the performance of network connections is known as *active queue management*. We are developing lightweight active queue management schemes for routers that (1) protect TCP connections from starvation by "aggressive" or non-responsive UDP applications, and (2) reserve router capacity for UDP applications that require continuous transmission. The goal is not to provide guarantees of performance or quality-of-service, but rather to provide a so-called "better-than-best-effort" forwarding service for multimedia flows that results in better throughput and latency for multimedia flows in times of congestion. The challenge is to provide a better-than-best-effort service without unnecessarily degrading TCP performance. We focus on a fundamentally best-effort service because we recognize that multimedia applications do not, in general, require or need performance guarantees. Most applications are capable of adapting to modest changes in network conditions but have minimal bandwidth and latency requirements that must be respected. Moreover, we are interested in determining how close one can come to achieving the performance of packet scheduling schemes with simpler and cheaper mechanisms.

The following section discusses active queue management in more detail and describes our approach. We also give some preliminary results showing that it is possible to meet our two goals with minimal modifications to existing active queue management schemes.

2. Active Queue Management

The default "best-effort" packet-forwarding service of IP is typically implemented in routers by a single, fixed-size, FIFO queue shared by all packets to be transmitted over an outbound link. The queue exists simply to provide capacity for tolerating variability ("bursts") in the arrival rate of packets. However, when the demand exceeds the available capacity of the outbound link for sustained periods of time, the queue capacity is exceeded. Router implementations using a simple fixed-size FIFO queue typically drop any packet that arrives at an already-full queue. This behavior is often called *drop-tail* packet discarding. Braden *et al.* describe two important problems with the drop-tail behavior [7]. First, in some situations, many of the flows can be "locked-out," a condition in which a small subset of the flows sharing the outbound link can monopolize the queue during periods of congestion. Flows generating packets at a high rate can fill up the queue such that packets from flows generating packets at substantially lower rates have a higher probability of arriving at the queue when it is full and being discarded. This has the effect of providing feedback to the "wrong" sources. The locked-out flows receive feedback that they should reduce their load while the "aggressive," high packet-rate flows receive proportionally less feedback and do not reduce their transmissions appropriately.

The second problem also occurs when the queue remains full or nearly full for sustained periods of time. When the queue is continually full, latency is increased for all flows. Simply making the queue shorter will decrease the latency but negates the possibility of accommodating brief bursts of traffic without dropping packets unnecessarily. For TCP-responsive flows, the latency problems associated with full queues can be addressed by selectively dropping packets *before* the queue fills. Since TCP flows decrease the load they generate in response to drops, dropping packets "early" should have the effect of eventually slowing or ceasing the growth of router queues. (How quickly this happens or if it happens at all depends on a variety of factors such as the round-trip latency for the individual flows.)

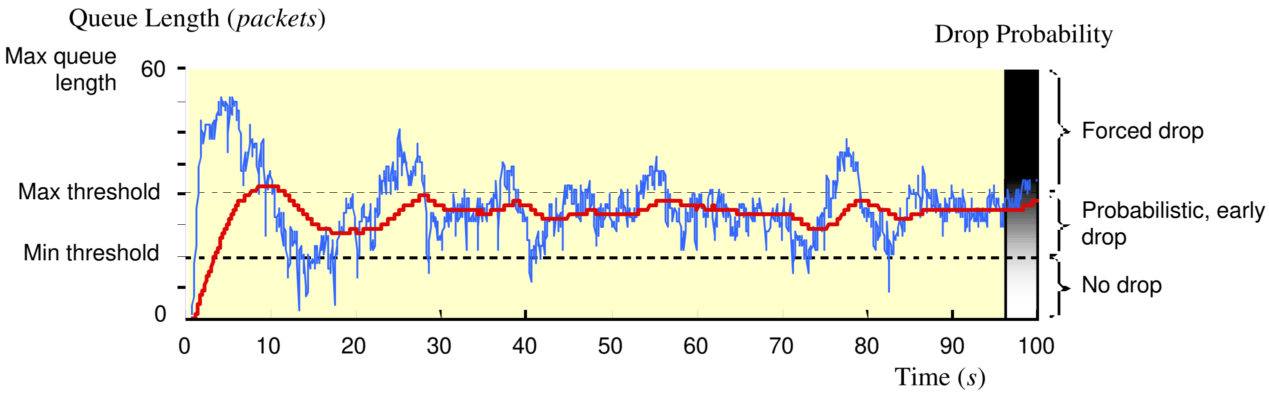

Figure 2: An illustration of the dynamics of the instantaneous queue length in a router and the RED weighted average queue length. The blue line represents the instantaneous queue length and the red line represents the weighted average queue length.

The Internet community is promoting a specific form of active queue management for routers known as *random early detection* or RED [7, 8]. Under RED, packets will randomly be dropped from a non-full queue with the probability of a packet being dropped at any given time being a function of the average length of the queue in the recent past.

At any time, a RED router operates in one of three modes. When a packet arrives at the router, a weighted average of the instantaneous queue length is computed. If the weighted average is less than a minimum threshold value, the router is considered to be uncongested and no drop action will be taken; the packet will simply be enqueued. This is the *no drop* mode of operation. If the weighted average queue length is greater than a minimum threshold value but less than a maximum threshold, this indicates some congestion has begun and some packets will be dropped probabilistically. In this case the router is in the *probabilistic drop* mode. In this mode an arriving packet is subjected to an *early drop test*. A random number is generated and compared to a computed drop probability threshold that is a function of the weighted average queue length and a drop probability parameter. To ensure that the router provides occasional feedback even when there is low-level but persistent congestion, a third parameter, the number of packets that have been enqueued since the last packet was dropped, is also used to compute the drop probability. If the random number generated is greater than the computed drop probability threshold, then the arriving packet is dropped. In the final case, if the average is greater than the maximum threshold value, a *forced drop* operation will occur. This is the *forced drop* mode, An average queue length in this range indicates persistent congestion and packets must be dropped to avoid a persistently full queue. The forced drop is also used in the special case where the queue is full but the average queue length is still below the maximum threshold (*i.e.*, in the case where the queue actually overflows but the long-term average queue length is less than the maximum threshold).

Figure 2 illustrates the queue length dynamics in a RED router. For the experiment illustrated in Figure 2, forced drops would occur only in the one short interval around time 10 when the weighted average reaches the maximum threshold. For most of the time the router is operating in the probabilistic drop mode.

By using a probabilistic mechanism for early drops, RED ensures that all flows experience the same percentage of packet loss. High-bandwidth flows will have a larger number of packets dropped since their packets arrive at a higher rate than lower-bandwidth flows (and thus are more likely to be subjected to a probabilistic, or forced drop), however, all flows experience the same loss rate. RED therefore avoids lockout by minimizing the probability of repeatedly penalizing the same flow when a burst of packets arrives. Moreover, by providing early notification of congestion, RED has been shown to result in shorter average queue lengths in routers [8].

Although RED improves the network performance of TCP flows, these flows are still subject to starvation in the presence of non-responsive UDP flows. In essence, the starvation dynamic described previously for a simple, non-RED FIFO queue, is unaffected by the imposition of RED. As TCP packets are dropped (for whatever reason), the TCP sources reduce their transmission rates while the non-responsive UDP flows occupy the queue elements previously occupied by TCP flows. The UDP flows experience the same loss rates as the TCP flows, that is, the UDP-based applications may perform quite poorly, but whatever performance levels they do achieve comes at the expense of TCP performance.

The starvation phenomenon has led to the call for routers to explicitly recognize non-responsive flows and to place them in a "penalty box" [9]. Non-responsive flows would be penalized by some draconian action such as having all their packets dropped during times of congestion. Such proposals are, on the one hand, seemingly necessary to protect TCP from non-responsive flows and to provide an incentive for applications to respond to packet loss as an indicator of congestion. On the other hand, the penalty box concept does not bode well for continuous media applications that while potentially adaptive, do not or can not adapt in a manner that mimics TCP's response to congestion.

We are experimenting with extensions to RED and other active queue management schemes that will isolate TCP flows from non-TCP flows while at the same time ensuring that continuous media applications receive a tunable share of network resources. We describe one such scheme, called *class-based thresholds* (CBT) [11], next.

3. Class-Based Thresholds — Active Queue Management for Multimedia Networking

Our approach is to isolate TCP flows from the effects of all other flows by constraining the average number of non-TCP packets that may reside simultaneously in a router's queue. We also want to isolate classes of non-TCP traffic from one another, specifically isolating certain continuous media traffic from all other traffic. To do this we tag packets of designated continuous media streams before they reach the router so that they can be classified appropriately. How this tagging is accomplished and the issues raised by tagging flows are outside the scope of this work, however, our operating assumption is that these flows are either self-identified at the end-system or identified by network administrators. For example, a tagging scheme using the *type-of-service* field in the IP header as part of a larger architecture for differentiated network services is one obvious possibility [10].

Once flows are tagged, a router will maintain simple queue statistics for a limited number of classes of traffic. At a minimum we assume the existence of a *TCP* class and a *marked* (*tagged*) *non-TCP* class. For simplicity, in the following discussion we assume that statistics are maintained for three traffic classes: *TCP*, *marked non-TCP*, and *all others*. The throughput of traffic classes will be constrained during times of congestion by limiting the average number of packets each traffic class can have enqueued in the router (thus limiting the fraction of link bandwidth each class can consume). Whenever a packet arrives at a router, it is classified into one of the three traffic classes. TCP packets are subject to the RED algorithm as described above. Thus if the average number of TCP packets in the queue remains less than the minimum RED threshold, an arriving TCP packet is guaranteed to be enqueued independent of the behavior of the other traffic classes.

For the non-TCP traffic classes, a weighted average number of enqueued packets from each class is maintained. When a non-TCP packet arrives, the weighted average for the appropriate class is updated and compared against a maximum threshold for the class. If the class average exceeds the threshold, the incoming packet is dropped. If the class average does not exceed the threshold then the packet is enqueued. Although packets are classified, there is still only one queue of packets, shared by all traffic classes, per outbound link in the router, and all packets are enqueued and dequeued in a FIFO manner.

In CBT, class thresholds only determine the ratios between the number of queue elements occupied by each class when all classes are fully utilizing the resources allocated to them. When one class is operating below its allocated queue capacity then other classes can effectively “borrow” that class’s unused link capacity. This happens because the total queue occupancy is reduced because one class is not using all of its allocated queue elements and, as a result, the average total queue length is reduced. When this occurs, packets from those classes that are operating at capacity, represent a larger fraction of the total packets enqueued and, thus, the classes operating at capacity are able to consume a larger fraction of the outbound link’s capacity. This is because ultimately, the link bandwidth consumed by a given traffic class is directly proportional to the average fraction of the queue that is occupied by packets of the class.

CBT provides isolation between traffic classes by maintaining separate thresholds for each class. These thresholds effectively allocate each class a portion of the queue’s capacity and ensure this capacity is available to the class independent of the transmission rates of other classes. Packets for a given class are dropped only if that class’s weighted average queue occupancy exceeds its allocation.

To demonstrate the effectiveness of CBT, we empirically compare the performance of CBT to simple FIFO (drop-tail) queuing and RED. We would naturally expect the worst performance for both TCP and continuous media UDP flows under FIFO. To understand what the best possible performance is, we also compare CBT to *class-based queuing* (CBQ), which is not a queue management scheme, but rather a packet scheduling scheme. In CBQ, link bandwidth (as opposed to queue elements) is explicitly allocated to traffic classes and packets are scheduled according to their allocation. Traffic classes are segregated into separate queues that are sampled according to their bandwidth allocations. In general, CBQ should provide optimal performance. Since traffic classes no longer share a common queue, all traffic classes are completely isolated from one another. If provisioned correctly by the CBQ allocation, continuous media flows should experience no drops and have minimal latency. While a packet scheduler such as CBQ can in fact provide performance guarantees to traffic classes, it is considerably more complicated to implement than an active queue management scheme for a FIFO queue. The interesting question is how closely an active queue management scheme such as CBT (or RED) can approximate the performance of CBQ.

In the following experiment FIFO, RED, CBT, and CBQ are implemented in a router. The router is configured with a 10 Mbps outbound link that is the bottleneck link in the network. In the CBT and CBQ implementations there are three classes of traffic: *TCP*, *marked non-TCP*, and *other* traffic. As before, the TCP traffic originates from a collection of users browsing the Web. In aggregate, these users are capable of utilizing 95% of a 10 Mb link. The marked non-TCP traffic originates from a set of end-systems that are participating in a set of six videoconferences each generating 220 *Kb/s* of UDP traffic.

Figure 3 illustrates the TCP throughput measurements on the outbound link of the router. Initially TCP's throughput is between 1,000 and 1,100 *KB/s* for each of the four methods. At approximately time 20, an aggressive and unresponsive high-bandwidth UDP flow is introduced and remains until time 70. It is the performance of the traffic classes during this "UDP blast" that highlights the benefits of CBT over the other queue management schemes. Under RED and FIFO, TCP throughput plummets to 200-400 *KB/s*. Throughput only remains as high as it does because of the sheer number of TCP flows in the experiment. In aggregate, the TCP flows maintain 200-400 *KB/s* even though individually some flows are suffering the starvation effect described above. Meanwhile, CBT and packet scheduling do show a decrease in TCP throughput, but the decrease is on the order of 10% for CBQ and 20% for CBT. This is because both methods constrain the amount of bandwidth that the aggressive flow can consume. TCP throughput decreases slightly because the aggressive flow is allowed a small fraction of the link's capacity.

In addition to isolating TCP in order to provide it with good performance, CBT also isolates the marked non-TCP multimedia flows from the non-responsive flows in the "other" class. This can be seen from the measured drop-rates for multimedia packets during the period of the UDP blast. These results are shown in Table 1. Both FIFO and RED produced extremely high drop-rates for multimedia while CBT produced drop-rates of just over 1%. No multimedia packets are lost under CBQ as it is tuned to meet the bandwidth requirements of the multimedia flows and as such, provides a guaranteed forwarding service to these applications. Note, however, that CBT closely approximates the performance of CBQ, and in

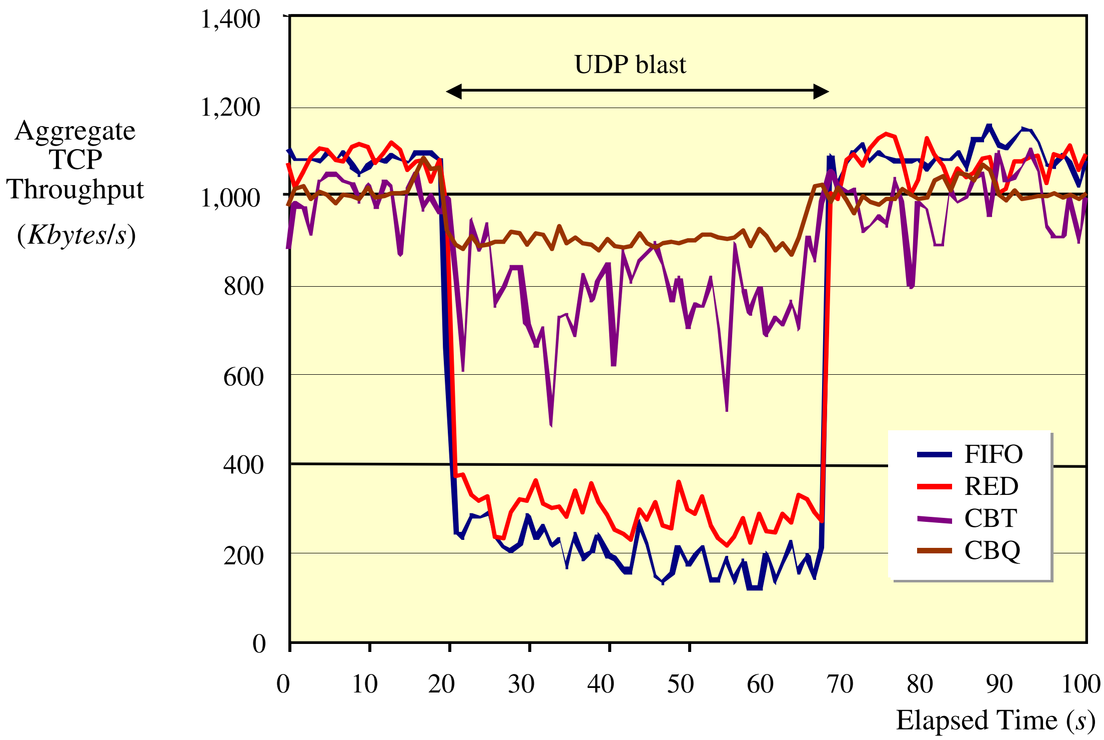

Figure 3: TCP throughput under each queue management/packet scheduling scheme.

particular, results in a drop rate for marked multimedia flows that is quite likely to be tolerated by the applications.

Table 1 also compares the average end-to-end network latency of marked multimedia packets. Latency is highest under simple FIFO queuing because the queue is consistently full and the average (successfully enqueued) arriving packet must wait for the entire queue to drain before it is transmitted. RED and CBT maintain shorter average queues, and this is reflected in their lower average latencies. Not surprisingly, CBQ provides the best average latency as it maintains a separate queue for packets from each traffic class. Under CBQ, applications such as videoconferencing that generate traffic at uniform rates and do not consume more bandwidth than has been provisioned for them, should experience minimal latency.

Table 1: Comparison of drop rates and network latency for tagged continuous media traffic under the various queue management alternatives.

Queue Management Scheme	**Packet Drop-rate for Marked Flows**	**Average Latency of Marked Flows**
FIFO	32.4%	63.2 *ms*
RED	30.0%	26.2 *ms*
CBT	1.3%	28.4 *ms*
Packet Scheduling (CBQ)	0.0%	5.7 *ms*

4. Summary

A salient requirement of interactive multimedia applications is that they transmit data continuously at uniform rates with minimum possible end-to-end delay. The majority of these applications do not require hard and fast guarantees of network performance, however, the current best-effort forwarding model of the Internet is frequently insufficient for realizing these requirements. Worse still, the requirement of uniform-rate transmission puts many multimedia applications at odds with current and proposed Internet network management practices which assume or require TCP-like reactions to packet loss. There exists a fundamental tension between the current practice of relying on end-systems to limit the rate at which they inject traffic into the network and the desire of many multimedia applications to transmit data continuously at a quasi-uniform rate. The Internet only works as well as it does because the vast majority of packets transmitted on the network are controlled by (loosely-speaking) the same congestion control and avoidance algorithm. Multimedia flows present a potential threat to this regime given their potentially high bandwidth requirements. Moreover, making these flows react to congestion in a TCP-like manner is not likely to work for all applications.

We are investigating a form of segregation between TCP and non-TCP continuous media flows. This will allow the research community to develop congestion control and avoidance mechanisms for continuous media applications that are inherently suited to the performance requirements of these applications without having to worry about compatibility with TCP. We have focused on simple router-based mechanisms such as active queue management rather than more complex mechanisms such as packet scheduling. In particular we are investigating the use of queue occupancy thresholds to isolate TCP flows from non-TCP flows (and vice versa) and to provide a better-than-best-effort forwarding service for flows that have been marked as being in need of this service. Our current scheme, called *class-based thresholds* (CBT), relies on a packet marking mechanism such as those proposed for realizing differentiated services on the Internet. CBT, when combined with existing active router queue management schemes such as RED, provides performance for TCP that approximates that achievable under a packet scheduling scheme, and acceptable performance for multimedia flows. CBT is a simple and efficient mechanism with implementation complexity and run-time overhead comparable to that of RED.

6. Acknowledgements

The research described here has been performed by the Distributed and Real-Time Systems group at the University of North Carolina at Chapel Hill (see URL *http://www.cs.unc.edu/Research/dirt*). Mark Parris is the inventor of the CBT concept and performed the experiments described in Section 3. Other team members and contributors include Don Smith, David Ott, Michele Clark, Jan Borgersen, Arun Moorthy, Mikkel Christiansen, Ramkumar Parameswaran, Gerardo Lamastra, Felix Hernandez, and Karim Mardini.

Support for this research has come from the National Science Foundation (grants CDA-9624662, CCR 95-10156, and IRIS-9508514), the National Institute of Health (National Center for Research Resources Award RR02170), the IBM Corporation, the Intel Corporation, Cabletron Incorporated, Advanced Networks & Services Inc., and the North Carolina Networking Initiative.

5. References

[1] National Tele-Immersion Initiative, Advanced Network and Services, Inc., *http://www.advanced.org/teleimmersion.html*, 1998.

[2] Taylor, R., *The nanoManipulator* (UNC-CH), *http://www.cs.unc.edu/Research/nano/*, November 1998.

[3] Talley, T.M., Jeffay, K., *Two-Dimensional Scaling Techniques for Adaptive, Rate-Based Transmission Control of Live Audio and Video Streams*, Proc. of ACM Multimedia '94, San Francisco, CA, Oct 1994, pp. 247-254.

[4] Delgrossi, L., *et al.*, *Media Scaling for Audiovisual Communication with the Heidelberg Transport System*, ACM Multimedia '93, Anaheim, CA, August 1993, pp. 99-104.

[5] Nee, P., Jeffay, K., Danneels, G., *The Performance of Two-Dimensional Media Scaling for Internet Videoconferencing*, Proceedings of the Seventh International Workshop on Network and Operating System Support for Digital Audio and Video, St. Louis, MO, May 1997, pages 237-248.

[6] Claffy, K., Miller, G., Thompson, K., *The nature of the beast: recent traffic measurements from an Internet backbone*, Proc., INET '98, Geneva, Switzerland, July 1998.

[7] Braden, B. *et al.*, *Recommendations on Queue Management and Congestion Avoidance in the Internet*, Internet draft, work in progress, 1998.

[8] Floyd, S., Jacobson, V., *Random Early Detection Gateways for Congestion Avoidance*, *IEEE/ACM Trans. on Networking*, Vol. 1, No. 4, August 1993, pp. 397-413.

[9] Floyd, S., Fall, K., *Promoting the Use of End-to-End Congestion Control in the Internet*, *Trans. on Networking*, Vol. 7, No. 4, August 1999, pp. 458-472.

[10] Nichols, K., Jacobson, V., Zhang, L., *A Two-bit Differentiated Services Architecture for the Internet*, Internet draft, work in progress, 1997.

[11] Parris, M., Jeffay, K.. Smith, F.D., *Lightweight Active Router-Queue Management for Multimedia Networking*, in *Multimedia Computing and Networking 1999*, Proceedings, SPIE Proceedings Series, Volume 3654, San Jose, CA, January 1999, pp. 162-174.