Dynamic-CBT – Better Performing Active Queue Management for Multimedia Networking

Mark Claypool and Jae Chung
Computer Science Department
Worcester Polytechnic Institute
Worcester, MA 01609
{claypool|goos}@cs.wpi.edu

Abstract

The explosive increase in Internet traffic has placed a growing emphasis on congestion control and fairness in Internet routers. Approaches to the problem of congestion, such as active queue management schemes like Random Early Detection (RED) use congestion avoidance techniques and are successful with TCP flows. Approaches to the problem of fairness, such as Fair Random Early Drop (FRED), punish misbehaved, non-TCP flows. Unfortunately, these punishment mechanisms result in a significant performance drop for multimedia flows that are well behaved. We extend Class-Based Threshold (CBT), and propose a new active queue management mechanism as an extension to RED called Dynamic Class-Based Threshold (D-CBT) to improve multimedia performance on the Internet. The performance of our proposed mechanisms is measured, analyzed and compared with other mechanisms (RED and CBT) in terms of throughput and fairness through simulation using NS. The study shows that D-CBT improves fairness among different classes of flows.

1. Introduction

The Internet has moved from a data communication network for a few privileged professions to an essential part of public life similar to the public telephone networks, while assuming the role of the underlying communication network for multimedia applications such as Internet phone, video conferencing and video on demand (VOD). As a consequence, the volume of traffic and the number of simultaneous active flows that an Internet router handles has increased dramatically, placing new emphasis on congestion control and traffic fairness. Complicating traditional congestion control is the presence of multimedia traffic that has strict timing constraints, specially *delay* constraints and variance in delay, or *jitter* constraints [1,2]. This paper presents a router queue management mechanism that addresses the problem of congestion and fairness, and improves multimedia performance on the Internet. Figure 1 shows some of the current and the proposed router queue mechanisms.

There have been two major approaches suggested to handle congestion by means other than traditional drop-tail FIFO queuing. The first approach uses packet or link scheduling on multiple logical or physical queues to explicitly reserve and allocate output bandwidth to each class of traffic, where a class can be a single flow or a group of similar flows. This is the basic idea of various Fair Queuing (FQ) disciplines and the Class-Based Queuing (CBQ) algorithm [3]. When coupled with admission control, the mechanism not only suggests a solution to the problem of congestion but also offers potential performance guarantees for the multimedia traffic class. However, this explicit resource reservation approach would change the "best effort" nature of the current Internet, and the fairness definition of the traditional Internet may no longer be preserved. Adopting this mechanism would require a change in the network management and billing practices. Also, the algorithmic complexity and state requirements of scheduling make its deployment difficult [4].

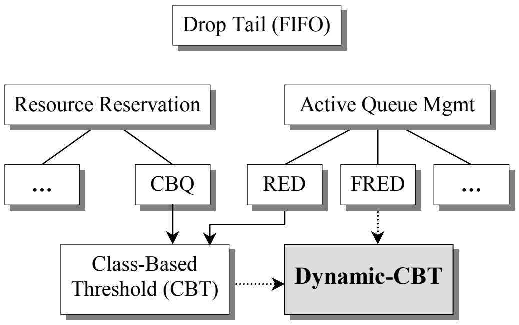


Figure 1: Router Queue Mechanisms (shaded is proposed)

The second approach, called Active Queue Management, uses advanced packet queuing disciplines other than traditional FIFO drop-tail queuing on an outbound queue of a router to actively handle (or avoid) congestion with the help of cooperative traffic sources. In the Internet, TCP recognizes packet loss as an indicator of network congestion, and its back-off algorithm reduces transmission load when network congestion is detected [5]. One of the earliest and well-known active queue management mechanism is Random Early Detection (RED), which prevents congestion through monitoring outbound buffers to detect impending congestion, and randomly chooses and notifies senders of network congestion so that they can reduce their transmission rate [6]. While fairly handling congestion for TCP flows, RED reveals the critical problem that non-TCP flows that are unresponsive or have greedier flow-control mechanisms than TCP can take more share of the output bandwidth than TCP flows [4,7]. In the worst case, it is possible for non-TCP flows, especially for unresponsive ones, to monopolize the output bandwidth while TCP connections are forced to transmit at their minimum rates. This unfairness occurs because non-TCP flows reduce transmission load relatively less than TCP flows or do not reduce at all, and the same drop rate is applied to every flow.

This unfairness could be a serious problem in a near future as the number of multimedia flows increases. Delay sensitive Internet multimedia applications typically use UDP rather than TCP because they require in-time packet delivery and can tolerate some loss, rather than the guaranteed packet delivery with potentially large end-to-end delay that TCP produces. Also, they prefer the periodic packet transmission characteristics of UDP rather than the bursty packet transmission characteristics of TCP that can introduce higher receiver side jitter. Multimedia UDP applications either do not use any flow-control mechanism or use their own application-level flow control mechanisms that are rate-based rather than window based and tend to be greedier than that of TCP taking the multimedia Quality of Service (QoS) requirements into account.

In addressing the problem of fairness, there have been strong arguments that unresponsive or misbehaving flows should be penalized to protect well-behaved TCP flows[1] [8]. Fair Random Early Drop (FRED) is an active queue management approaches that incorporate this argument [7]. FRED adds per-active-flow accounting to RED, isolating each flow from the effect of others. It enforces fairness in terms of output buffer space by strictly penalizing unresponsive or misbehaving flows to have an equal fair share while assuring packets from flows that do not consume their fair share are transmitted without loss. FRED serves its purpose not only in protecting TCP flows from unresponsive and misbehaving flows but also in protecting fragile TCP connections from robust TCP connections. However, the per-active-flow accounting is expensive and might not scale well. FRED also has a potential problem that its TCP favored per-flow punishment could unnecessarily discourage flow-controlled interactive multimedia flows. Under FRED, incoming packets for a well-behaved TCP flow consuming more than their fair share are randomly dropped applying RED's drop rate. However, once a flow, although flow controlled, is marked as a non-TCP friendly flow, it is regarded as an unresponsive flow and all incoming packets of the flow are dropped when it is using more than its fair share. As a result, a flow-controlled multimedia UDP flow, which may have a higher chance to be marked, will experience more packet loss than a TCP flow and be forced to have less than its fair share of bandwidth.

Jeffay *et al.,* [4] propose a new active queue management scheme called Class-Based Threshold (CBT), which releases UDP flows from strict per-flow punishment while protecting TCP flows by adding a simple class-based static bandwidth reservation mechanism to RED. In fact, CBT implements an explicit resource reservation feature of CBQ on a single queue that is fully or partially managed by RED without using packet scheduling. Instead, it uses class thresholds that determine ratios between the number of queue elements that each class may use during congestion. CBT defines three classes: tagged (multimedia) UDP[2], untagged (other) UDP and TCP. For each of the two UDP classes, CBT assigns a pre-determined static threshold and maintains a weighted-average number of enqueued packets that belong to the class, and drops the incoming class' packets when the class average exceed the class threshold. By applying a threshold test to each UDP class, CBT protects TCP flows from unresponsive or misbehaving UDP flows, and also protects multimedia UDP flows from the effect of other UDP flows. CBT avoids congestion as well as RED, has less overhead and improves multimedia throughput and packet drop rates compared to FRED. However, as in the case of CBQ, the static resource reservation mechanism of CBT could result in poor performance for rapidly changing traffic mixes and is arguably unfair since it changes the best effort nature of the Internet.

To eliminate the limitations due to the explicit resource reservation of CBT while preserving its good features from class-based isolation, we propose *Dynamic-CBT (D-CBT).* D-CBT fairly allocates the bandwidth of a congested link to the traffic classes by dynamically assigning the UDP thresholds such that the sum of the fair share of flows in each class is assigned to the class at any given time.

We use an event driven network simulator called NS (version 2) [9] to evaluate D-CBT. NS implements most of common IP network components including RED. We implement CBT in NS, extend it to D-CBT, and compare the performance of D-CBT with that of RED and CBT. In the evaluation, our primary focus is on the effect of heterogeneously flow-controlled traffic on the behavior of the queue management mechanisms especially on fairness.

Section 2 discusses CBT and Section 3 presents D-CBT in detail. Section 4 describes our simulation setup, Section 5 analyzes and evaluates D-CBT and Section 6 concludes our research.

2. Class-Based Threshold (CBT)

Before describing D-CBT, we briefly discuss the design of Class-Based Threshold (CBT) [4] which D-CBT extends. As discussed briefly in Section 1, the main idea behind the design of CBT is to apply class-based isolation on a single queue that is fully or partially managed by RED without using packet scheduling. Instead of using packet scheduling on multiple logical queues, CBT regulates congestion-time output bandwidth for *n* classes of flows using a RED queue management mechanism and a threshold for each of the *n-1* classes of flows, which is the average number of queue units that a class may use.

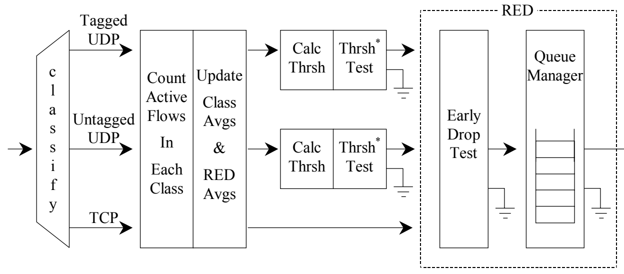
CBT categorizes flows into three classes, which are TCP, tagged (multimedia) UDP and untagged (other) UDP, and assigns a pre-determined static threshold for each of the two UDP classes, assuming that UDP flows are mostly unresponsive or misbehaving and need to be regulated. When a UDP packet arrives, the weighted-average for the appropriate class is updated and compared against the threshold for the class to decide whether to drop the packet before passing it to the RED algorithm. For the TCP class, CBT does not apply a threshold test but directly passes incoming packets to the RED test unit. This is the first design of CBT, called "CBT with RED for all". In the second design, called "CBT with RED for TCP", only TCP packets are subjected to RED's early drop test, and UDP packets that survive a threshold test are directly enqueued to the outbound queue that is managed by RED. Another difference from the first design is that RED's average queue size is calculated only using the number of enqueued TCP packets. CBT with RED for TCP is based on the assumption that tagged (multimedia) UDP flows as well as untagged (other) UDP flows are mostly unresponsive, and it is of no use to notify these traffic sources of congestion earlier. D-CBT is extended from CBT with RED for all. In the rest of this paper, CBT refers to CBT with RED for all.

[1] A *well-behaved flow* (or TCP friendly) is defined as a flow that behaves like a TCP flow with a correct congestion avoidance implementation. A flow-controlled flow that acts different (or greedier) than well-behaved flow is a *misbehaving flow*.

[2] Tagged (multimedia) UDP flows can be distinguished from other (untagged) UDP flows by setting an unused bit of the Type of Service field in the IP header (Version 4).

3. Dynamic-CBT (D-CBT)

D-CBT enforces fairness among classes of flows, and gives UDP classes better queuing resource utilization. Figure 2 shows the design of D-CBT. The key difference from CBT is (1) the dynamically moving fair thresholds and (2) the UDP class threshold test that actively monitors and responds to RED indicated congestion. To be more specific, by dynamically assigning the UDP thresholds such that the sum of the fair average queue resource share of flows in each class[3] is assigned to the class at any given time, D-CBT fairly allocates the bandwidth of a congested link to the traffic classes. Also, the threshold test units, which are activated when RED declares impending congestion (i.e. *red_avg* > *red_min*), coupled with the fair class thresholds, allow the UDP classes to use the available queue resources more effectively than in CBT, in which each UDP class uses the queue elements an average of no more than its fixed threshold at any time. Looking at it from a different view, D-CBT can be thought of a Class-Based FRED-like mechanism that does *per-class-accounting* on the three classes of flows.



* Threshold Test is activated when *red_avg* > *red_min*

Figure 2: Design of Dynamic-CBT (D-CBT)

As in CBT, D-CBT categorizes flows into TCP, tagged UDP and untagged UDP classes. However, unlike the class categorization of CBT in which flow-controlled multimedia flows are not distinguished from unresponsive multimedia flows (all tagged), D-CBT classifies UDP flows into flow-controlled multimedia (tagged) UDP and other (untagged) UDP. The objective behind this classification is to protect flow-controlled multimedia flows from unresponsive multimedia flows, and encourage multimedia applications to use congestion avoidance mechanisms, which may be different than those of TCP. We believe that there are advantages in categorizing UDP traffic in this way for the following reasons: First, multimedia applications are the primary flows that use high bandwidth UDP. Second, by categorizing flows by their congestion responsiveness characteristic (i.e. TCP friendly, flow-controlled but misbehaving multimedia and unresponsive flows), different management can be applied to the classes of differently flow-controlled flows.

In fact, in determining the fair UDP thresholds, D-CBT calculates the fair average output buffer share of the tagged UDP class from the average queue length that is maintained by RED, and that of untagged UDP class from the RED's minimum threshold (plus a small allowance). This is based on the assumption that tagged flows (or flow-controlled multimedia) can respond to network congestion and will actively try to lower the average length of a congested queue on notification of congestion. Therefore, they are allowed to use the impending congestion state queue buffers (i.e. *red_avg* – *red_min* when *red_avg* > *red_min*) up to their fair share of the average. However, unresponsive (untagged) flows, which have no ability to respond to network congestion, are not allowed to use the impending state queue buffers at impending congestion. Actually, we allow the unresponsive UDP class to use a small fraction of the impending state queue buffers, which is 10% of (*red_max* – *red_min*) * *untagged_UDP_share* when the maximum early drop rate is 0.1, to compensate for the effect of needless additional early drops for the class.

In the design of D-CBT, the existence of the active flow counting unit is a big structural difference from CBT. In order to calculate a fair threshold (or average queue resource share) for each class, D-CBT needs class state information, and therefore keeps track of the number of active flows in each class. Generally, as in FRED, active flows are defined as ones whose packets are in the outbound queue [7]. However, we took slightly different approach in detecting active flows, in that an active flow is one whose packet has entered the outbound queue unit during a certain predefined interval since the last time checked. In D-CBT, an active flow counting unit that comes right after the classifier maintains a sorted linked list, which contains a flow descriptor and its last packet reception time, and a flow counter for each class. Currently, the flow descriptor consists of a destination IP address and the flow ID (IPv6). However, assuming IPv4, this could be replaced by source and destination address, although this would redefine a flow as per source-destination pair.

For an incoming packet after the classification, the counting unit updates an appropriate data structure by inserting or updating the flow information and the current local time. When inserting new flow information, the flow counter of the class is also increased by one. The counting unit, at a given interval (set to 300ms in our implementation), traverses each class' linked list, deletes the old flow information and decreases the flow counter. The objective behind this probabilistic active flow counting approach is twofold: First, D-CBT does not necessarily require an exact count of active flows as do other queue mechanisms that are based on flow-based-accounting, although a more exact count is better for exercising fairness among flow classes. Second, it might be possible to improve the mechanism's packet processing delay by localizing the counting unit with the help of router's operating system and/or device. For example, the traversing delete is a garbage collection-like operation that could be performed during the router's idle time or possibly processed by a dedicated processor in a multiprocessor environment. In our implementation, we used a sorted linked list data structure that has the inserting and updating complexity of $O(n)$, and the traversing complexity of $O(n)$, where n is the number of flows of a class. Assuming that a simple hash table is used instead, the complexity of inserting and updating operation drops to $O(1)$, while the complexity of the traverse delete will remain $O(n)$.

When an incoming packet is updated or inserted according to its flow identification to its class data structure at the counting unit, D-CBT updates the RED queue average, the tagged UDP average and the untagged UDP average, and passes the packet to an appropriate test unit as shown in Figure 2. Note that for every incoming packet all of the averages are updated using the same weight. This is to apply the same updating ratio to the weighted-

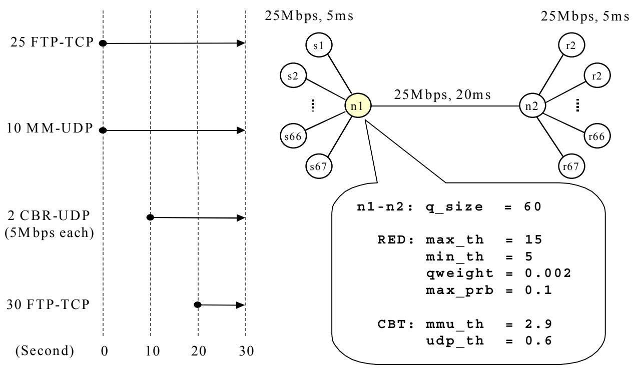
[3] Fair class shares are calculated based on the ratio between the number of active flows in each class.

averages, so that a snapshot in time at any state gives the correct average usage ratio among the classes. Using the three averages and the active flow count for each class, the UDP threshold test units calculate the fair thresholds for the tagged and untagged UDP classes, and apply the threshold test to incoming packets of the class when the RED queue indicates impending congestion. UDP packets that survive an appropriate threshold test are passed to the RED unit along with the TCP flows as in CBT.

Thus, D-CBT is designed to provide traditional fairness between flows of different characteristics by classifying and applying different enqueue policies to them, and restrict each UDP class to use the queue buffer space up to their share in average. We hypothesize that the advantages of D-CBT are the following: First, D-CBT avoids congestion as well as RED with the help of responsive traffic sources. Second, assuming that the flows in a class (especially the tagged UDP) use flow control mechanisms of which the congestion responsiveness characteristics are almost the same, D-CBT will fairly assign bandwidth to each flow with much less overhead than FRED, which requires per-flow state information. Even if the tagged flows do not use their fair share, D-CBT will still successfully assign bandwidth fairly to each class of flows, protecting TCP from the effect of misbehaving and unresponsive flows and also protecting the misbehaving (flow-controlled multimedia) flows from the effect of unresponsive flows. Lastly, D-CBT gives tagged (flow-controlled multimedia) flows a better chance to fairly consume the output bandwidth than under FRED by performing per-class punishments instead of the strict per-flow punishment.

4. Simulation

We ran a simulation for each of RED, CBT and D-CBT. Every simulation had the exactly same settings except for the network routers, each of which was set to use one of the above three outbound queue management mechanisms. The network topology and the traffic source schedules are shown in Figure 3.



```
n1-n2: q_size   = 60

  RED: max_th   = 15
       min_th   = 5
       qweight  = 0.002
       max_prb  = 0.1

  CBT: mmu_th   = 2.9
       udp_th   = 0.6
```

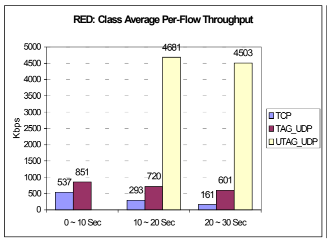
Figure 3: Simulation Scenario and Network Setup

For traffic sources, 55 FTP, 10 flow-controlled multimedia traffic generator called MM_APP [10] (tagged) and 2 CBR (untagged) traffic generators were used, where FTP used TCP Reno and the others used UDP as the underlying transport agent. The network packet size was set to 1Kbyte. The MM_APP traffic generators, which react to congestion using 5 discrete media scales with a "cut scale by half at frame loss, up scale by one at RTT" flow control mechanism, used 300, 500, 700, 900 and 1,100Kbps for scale 0 to 4 transmission rates, with a fixed frame size of 1Kbyte. The CBR sources were set to generate 1Kbyte packets at a rate of 5Mbps.

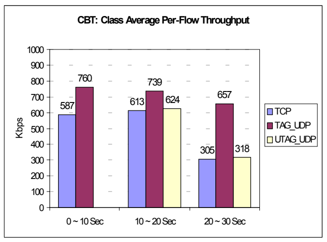
Network routers were assigned a 60-packet long physical outbound queue. The RED parameters, which are shown in Figure 3, were chosen from one of the sets that are recommended by Floyd and Jacobson [11]. For CBT, beside the RED parameters, the tagged and untagged class thresholds (denoted as *mmu_th* and *udp_th* in the figure) were set to 2.9 packets and 0.6 packets to force each UDP to get about their fair bandwidth shares during 0 to 20 seconds. D-CBT also shares the RED settings, however, since each threshold is assigned dynamically to the fair share of each class, no threshold setup was necessary.

5. Result and Analysis

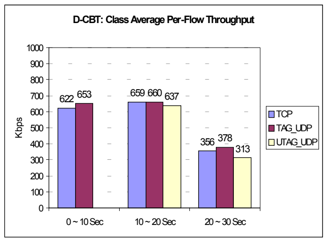
We measured the performance of RED, CBT and D-CBT in terms of fairness. In this section, we compare *the average per-flow throughput in each class*, which is an average aggregated class throughput divided by the number of flows in the class, to visualize how fairly the output bandwidth is assigned to each class considering the number of flows in the class. Figure 4 (a) through (c) compares the periodic (i.e., 0-10, 10-20 and 20-30 seconds) average per-flow throughput for each class under the three queue mechanisms.



(a) RED



(b) CBT



(c) D-CBT

Figure 4: Average Per-Flow Throughput for TCP, Tagged UDP and Untagged UDP Classes under RED, CBT and D-CBT

As shown in Figure 4 (a), RED absolutely failed to assign bandwidth fairly to each class of flows from 10 seconds when the two high bandwidth untagged UDP flows (unresponsive CBR) join transmitting at the total of 10Mbps, about 40% of the link bandwidth. During 0-10 seconds, when 25 TCP and 10 tagged (flow-controlled MM_APP) flows are competing for the bandwidth, it was somewhat unfair as a tagged flow gets an average of 37% more bandwidth than a TCP flow, but RED was able to manage the bandwidth. However, when the untagged UDP blast came into the system, RED was totally unable to manage bandwidth. The 2 untagged UDP flows got most of the bandwidth they needed (average of 4.68Mbps out of 5Mbps), and the remaining flows used the leftover bandwidth. Especially, the 25 TCP flows got severely punished and transmitted at an average of 293Kbps per flow as they often went back to slow start and even got timed out. Fairness got worse as 30 more TCP flows joined at 20 seconds, and experienced starvation.

Figure 4 (b) shows that CBT can avoid the great unfairness of RED using fixed thresholds for the UDP classes. CBT that uses a fixed threshold on UDP classes was able to avoid extreme unfairness. However, during the analysis, we found that CBT, which updates each class average and the RED average independently, suffers from *unsynchronized weighted-average updates*. That is, the ratio between independently updated UDP class averages and RED average does not correctly indicate the actual class bandwidth utilization ratio, since whichever flows updates the average more frequently will have higher weighted-average than the others will, although they all use the same amount of bandwidth. This is because as the average is updated more frequently, not only is a newly enqueued packet added to the average with a predetermined weight, but also the existence of the other already enqueued packet are added to the average.

Figure 4 (c) shows the D-CBT results, which indicates that D-CBT fairly manages bandwidth during all periods by dynamically allocating the right amount of output queue space to each flow class. It also shows that by updating each class and RED average at the same time in a synchronized manner, the ratio between the averages is a good indicator of the ratios between each class' bandwidth utilization. One thing to note in the figure is that although we strictly regulate the untagged class by assigning a fair threshold calculated from RED's minimum threshold, the untagged class did get most of its share. This is because the high bandwidth untagged (unresponsive) packets were allowed to enter the queue without a threshold test, when RED indicated no congestion.

6. Conclusion

In this paper, we have presented the design and evaluation of our proposed router queue mechanisms, Dynamic Class-Based Threshold (D-CBT), by comparing their performance with that of RED and CBT. D-CBT is a new active queue management mechanism that addresses the problem of fairness by grouping flows into TCP, tagged (flow-controlled multimedia) UDP and untagged (other) UDP classes and regulating the average queue usage of the UDP classes to their fair shares.

As expected, RED, that has shown to be fair among TCP flows, showed an extreme unfairness with mixed traffic. CBT that uses a fixed threshold on UDP classes was able to avoid extreme unfairness. However, during the analysis, we found that CBT suffers from "unsynchronized weighted-average updates". D-CBT fixes CBT's problem by synchronizing all the average updates, and better manages bandwidth by dynamically determining the UDP thresholds to cooperate with RED by fairly assigning the output bandwidth to each class for all traffic mixes. That is, through the class-based accounting, D-CBT fairly protects TCP from the effect of UDP flows and also fairly protects tagged UDP flows from untagged flows.

There exist many possible areas for future work and still remain many performance aspects to be evaluated. A study that we could not do due to the lack of time but suggest as a future work is to compare the performance of the D-CBT with that of FRED. We expect that D-CBT could give better throughput performance for tagged UDP flows than FRED, since it frees flow-controlled multimedia flows from the strict per-flow punishment.

7. References

[1] Multimedia Communications Forum, Inc. "Multimedia Communications Quality of Service", *MMCF/95-010, Approved Rev 1.0*, 1995, URL: http://www.luxcom.com/library/2000/mm_qos/qos.htm

[2] Claypool, M. and Tanner, J., "The Effects of Jitter on the Perceptual Quality of Video", *ACM Multimedia Conference*, Volume 2, Orlando, FL, October 30 - November 5, 1999

[3] Floyd, S. and Jacobson, V., "Link-sharing and Resource management Models for Packet Networks", *IEEE/ACM Transactions on Networking*, Vol. 3 No. 4, August 1995

[4] Parris, M., Jeffay, K. and Smith, F. D., "Lightweight Active Router-Queue Management for Multimedia Networking", *Multimedia Computing and Networking*, SPIE Proceedings Series, Vol. 3020, San Jose, CA, January 1999

[5] Floyd, S., "TCP and Explicit Congestion Notification", *Computer Communication Review*, October 1994

[6] Floyd, S. and Jacobson, V., "Random Early Detection Gateways for Congestion Avoidance", *IEEE/ACM Transactions on Networking*, August 1993

[7] Lin, D. and Morris R., "Dynamics of Random Early Detection", In *Proceedings of SIGCOMM '97*, Cannes, France, September 1997

[8] Floyd, S. and Fall, K., "Promoting the Use of End-to-End Congestion Control in the Internet", *IEEE/ACM Transactions on Networking*, February 1998

[9] VINT, "Virtual InterNetwork Testbed, A Collaboration among USC/ISI, Xerox PARC, LBNL, and UCB", URL: http://netweb.usc.edu/vint

[10] Chung, J. and Claypool, M., "Better-Behaved, Better-Performing Multimedia Networking", *SCS Euromedia Conference*, Antwerp, Belgium, May 8-10, 2000

[11] Raghavendra, A. M. and Kinicki, R. E., "A Simulation Performance Study of TCP Vegas and Random Early Detection", *IEEE International Performance, Computing, and Communications Conference 1999* (*IPCCC99)*, February 1999