Proportional Bandwidth, Delay, and Loss Differentiation

Manimaran Selvaraj
ScImage, Inc,
Los Altos, CA
mani@scimage.com

Georgios Y. Lazarou
TITL
Mississippi State University
glaz@ece.msstate.edu

Rose Hu
TITL
Mississippi State University
hu@ece.msstate.edu

ABSTRACT
A robust adaptive scheduler for proportional delay differentiation services is presented. Proportional services are further policed by a class based packet dropper. Our proposed combination of the adaptive scheduler and the packet dropper treats the traffic classes proportionally in terms of three QoS metrics: bandwidth, delay, and packet loss, simultaneously. Simulation experiments with bursty traffic validate our claim.

KEY WORDS
QoS, Proportional DiffServ, Scheduling

1 Introduction

Many Internet service providers (ISP) are now beginning to make use of techniques that differentiate one user application from the other. This relative differentiation is necessary due to several issues such as QoS provisioning and pricing policies. In Relative Differentiated Services a traffic class is treated *relative* to another traffic class. In this approach, Internet traffic is grouped into N finite number of classes. *Class i* gets a better or at least no worse service than *class i-1*. This is achieved through the use of packet schedulers and packet accept/discard rules.

The treatment meted out to the applications can be differentiated in terms of one or a combination of the three performance metrics: bandwidth, delay and packet loss. From an application point of view, traffic ought to be differentiated in terms of the delay and packet loss, since they differ in delay and loss requirements. Here, proportional delay or loss differentiation is most suitable. From an ISP's point of view, users have to be allocated bandwidth according to the service agreement. Thus, bandwidth differentiation is needed in this case.

Several scheduling algorithms [1][2][3][4][5] were proposed to achieve relative differentiation. However, all these algorithms provide a relative service differentiation in terms of only one of the three peformance metrics. Although, the work in [1], presents schemes to achieve a proportional delay as well as loss differentiation, a study of co-existence of the loss and delay differentiation schemes was not performed. Several of the schemes were also computationally demanding.

In this work we achieve a proportional bandwidth, delay, and loss differentiation simultaneously. The packet delay and loss are controlled to achieve a relative bandwidth, delay, as well as loss differentiation, all three at the same time. While controlling the delay by means of an adaptive version of the HPD scheduler [1], the packet loss is taken care of by a class based RED packet dropper. We achieve a proportional bandwidth differentiation by selectively dropping and delaying packets of different classes.

The remainder of the paper is organized as follows. Section 2 describes the previous work on the proportional delay and bandwidth differentiation. The proportional delay mechanism and the proportional bandwidth mechanism are discussed in sections 3 and 4 respectively. In section 5, we evaluate the performance of the proposed scheme, and finally section 6 concludes the paper.

2 Related Work

In the Proportional Delay Differentiation (PDD)[1] model, the ratio of the overall long term average delay, $\overline{d}$, experienced by two different traffic classes i and j is equal to the ratio of their corresponding delay differentiation parameters or class weights q:

$$\frac{\overline{d}_i}{\overline{d}_j} = \frac{q_i}{q_j} \quad (i, j = 1...N) \tag{1}$$

The class weights $\{q_i\}$, are ordered such that the higher classes experience less delay than the lower classes.

The work in [1] proposed the use of three packet schedulers to achieve proportional delay differentiation. The schedulers are the proportional average delay scheduler (PAD), the waiting time priority scheduler (WTP), and the hybrid proportional delay scheduler (HPD). The HPD scheduler was an attempt to design a packet scheduler which had the best features of both PAD and WTP. The normalized average delay of the HPD is given as:

$$\tilde{h}_i(t) = (g)\tilde{d}_i(t) + (1-g)\tilde{w}_i(t) \tag{2}$$

where *'g'* is the HPD parameter, $\tilde{w}_i(t)$ is the normalized head packet waiting time as calculated by WTP [1], and $\tilde{d}_i(t)$ is the normalized delay as calculated by PAD [2]. The HPD parameter *'g'* plays a very significant role in HPD's performance. Under heavy loads, the value of *'g'* does not affect the performance of HPD, since both PAD and WTP work

[1] average of waiting times of first packet in queue
[2] average of delay experienced by all dequed packets

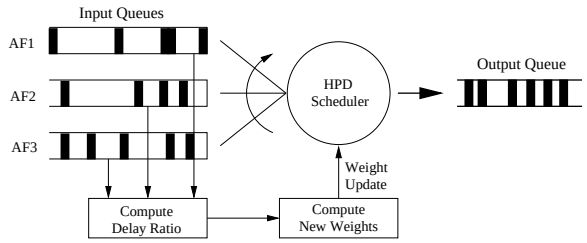

Figure 1. Adaptive HPD

well under heavy loads. But at lower utilization, *'g'* must be set close to 1 so that HPD works more like PAD [1]. The value of *'g'* is set to 0.85 in simulation experiments. In general, the WTP and HPD schedulers perform better than the PAD. PAD is able to meet the PDD model only when the delay differentiation parameters are available; whereas, WTP works only under heavy loads.

A relative bandwidth differentiation between TCP micro-flows was achieved in [6] by making use of the weighted version of RED, called WRED [7]. In [6], WRED was used to achieve a per-flow relative loss differentiation. They further proposed that a relative bandwidth differentiation could be achieved by a combination of a relative loss and a relative delay differentiation of the TCP micro-flows. Unlike [6], we achieve a relative bandwidth differentiation not between micro-flows, but between aggregates. This Proportional Bandwidth Differentiation model is the bandwidth analogy of the Proportional Delay Differentiation Service model [1].

3 Proportional Delay Mechanism

Motivated by the works in [2] and [3], we made the HPD packet scheduler [1] adaptive, so that the scheduler maintains the desired delay differentiation ratio under realistic bursty network traffic conditions. The *adaptiveness* in the HPD scheduler helped to maintain the desired proportional delay differentiation. Unlike other adaptive approaches [2] [3], the adaptive approach proposed here is much simpler, and it does not depend on the network load.

3.1 Adaptive HPD

The foremost work [1] on the proportional delay differentiation model gave examples of the environments in which the model worked. But, the schedulers mentioned in [1] did not achieve a proportional delay differentiation under low and medium system utilization. We propose a major addendum to the HPD scheduling scheme by adding a feedback component to the scheduler. We call this the adaptive HPD (AHPD) scheduler and is depicted in Fig. 1. In this proposed new scheduler, the actual delay ratio between two traffic classes is periodically monitored and the class weights are changed in such a way that the actual delay ratio is always maintained at the desired value.

Let D_i be the corresponding average end-to-end delay experienced by the AF classes AF_i. Let the delay differentiation ratios be $\Delta_i = \frac{D_i}{D_{i+1}}$. The overall average delay D_i is inversely related to the normalized delay $\tilde{h}_i(t)$, because the scheduler serves the queue with the maximum normalized delay, i.e., if $\tilde{h}_i(t)$ gets higher, then that particular queue is more likely to be served and the corresponding overall delay D_i will be reduced. Hence, the delay differentiation ratios can then be represented as:

$$\Delta_i = \frac{D_i}{D_{i+1}} \approx \frac{\tilde{h}_{i+1}(t)}{\tilde{h}_i(t)}, \tag{3}$$

Our scheduler works to maintain the delay differentiation ratios Δ_i at a desired level by varying the AF class weights q_i.

Whenever a packet is served, Δ_i is computed using Equation (3). If the scheduler delay differentiates perfectly, Δ_i will always be equal to a desired (constant) value K. So, if the delay ratio falls inside a window $(K-\varepsilon, K+\varepsilon)$ around the desired value K, the scheduler parameters are left unchanged. The scheduler adjusts the weights whenever the delay differentiation ratio Δ_i deviates from its corresponding window $(K-\varepsilon, K+\varepsilon)$.

The weights are changed according to the weight function:

$$f(q_i) = \begin{cases} q_i = q_i + \Psi & \& \quad q_{i-1} = q_{i-1} - \Psi & for & K < 2-\varepsilon \\ q_i = q_i^{init} & \& \quad q_{i-1} = q_{i-1}^{init} & for & K-\varepsilon < \Delta_i < K+\varepsilon \\ q_i = q_i - \Phi & \& \quad q_{i-1} = q_{i-1} + \Phi & for & \Delta_i > K+\varepsilon \end{cases} \tag{4}$$

where q_i^{init} is the initial value of weight i. This function was formulated based on the property that decreasing/increasing the weight of a class affects the average delay of all other classes as well as its own average delay [1]. In the simulation experiments, ε was set to 0.25. ε is set based on a tradeoff between the number of computations and the stringent maintenance of the delay ratio. Setting ε = 0 would result in weight update computations upon every packet arrival, while a very higher value of ε (say $\varepsilon > 1$), would result in a performance similar to the original HPD scheme. In (4), Ψ and Φ are calculated as:

$$\Psi = \frac{(q_{max}^i - q_{curr}^i) \times |(K-\varepsilon) - \Delta_i|}{(q_{max}^i - q_{min}^i)}$$

$$\Phi = \frac{(q_{max}^i - q_{curr}^i) \times |\Delta_i - (K+\varepsilon)|}{(q_{max}^i - q_{min}^i)}$$

where q_{max}^i and q_{min}^i are the maximum and minimum possible values of a class weight respectively, and q_{curr}^i is the current value of the weight in a cycle. $|\Delta_i - (K \pm \varepsilon)|$ represents the deviation of the computed delay differentiation ratio Δ_i from the window $(K-\varepsilon, K+\varepsilon)$. $(q_{max}^i - q_{min}^i)$ is the range of weights and $(q_{max}^i - q_{curr}^i)$ is the maximum value by which the current value of a weight can be increased or decreased. Ψ and Φ were designed in such a way that deviations in the delay ratios are corrected by an increase or decrease of the weights in a linear fashion.

The class weights are initially set such that the ideal delay ratio is obtained. For example, when the initial ideal weight values are set as q_1 = 1, q_2 = 2, and q_3 = 4, the desired delay ratio, which must be proportional to the ratio of weights, is $\Delta_i = \frac{q_{i+1}}{q_i} = K = 2$. Table 1 gives the corresponding maximum and minimum weights for each of the AF classes. The maximum (minimum) value of a particular weight is computed as the average of the weight's initial value and the initial value of the next higher (lower) weight.

Table 1. Maximum and minimum weights for each class.

Class	Initial Weight	Maximum Weight	Minimum Weight
AF1	q^0_{init} = 1	q^0_{max} = 1.5	q^0_{min} = 0.5
AF2	q^1_{init} = 2	q^1_{max} = 3	q^1_{min} = 1.5
AF3	q^2_{init} = 4	q^2_{max} = 6	q^2_{min} = 3

The action taken by the proposed AHPD scheduler when packets arrive is described in the following:

1. Initialization. Set the initial parameters q_i, g, and the desired delay differentiation ratio Δ_i. Compute initial $\tilde{h}_i$. When no packet has been served, select the queue to start service using the initial weights q_i.
2. Whenever a queue is served, update the parameters as follows.
   (a) Calculate new Δ_2 using equation (3).
   (b) Update the weights q_3 and q_2 using equation (4).
   (c) Calculate new Δ_1 using equation (3).
   (d) Update weight q_1 using equation (4).
   (e) Compute the new normalized average delay by using equation (2).
3. Select the queue with the maximum normalized average delay and serve that queue.
4. Save the updated weights for the next cycle.
5. Go back to 2.

To summarize, we first update the weights of the two highest priority classes. Then the weight of the next lower priority class is updated based on the weight of its predecessor.

4 Proportional Bandwidth Mechanism

Based on [6], we propose the use of a RIO-like [8] packet marking and dropping scheme, where the RED drop probabilities of the different classes are proportional to each other. But, we achieve a per-aggregate bandwidth differentiation. In our scheme, the edge router upon receiving a packet, marks it as either green, yellow or red colored packet based on the packet's class. Thus, all the flows within a class are colored same. The packets with different colors experience different accept/discard treatment. We call our scheme colored-RED (CRED), since the RED parameters are not the same for the different colors.

In the simulation experiments, we applied CRED on a system with 3 AF classes, each with 2 drop precedences. In CRED, packets belonging to AF3, AF2, and AF1 classes are colored green, yellow, and red respectively. The drop probability for AF3 class is the smallest of the three. The average queue size and the maximum drop probability are calculated in such a way that various classes are proportionally treated. The maximum drop probability, max_p, of the different AF classes is fixed in the same way as in [6]. The max_p values are set as: max_p(red) = BDP * max_p(yellow) and max_p(yellow) = BDP * max_p(green), where BDP is the bandwidth differentiation parameter.

A major difference between CRED and [6] is that, in CRED, the average queue size (AQS) calculation of a class is independent of the other class packets. In [6] the average queue size is computed as:

AQS_{AFi} = TSW estimate based on (AF1 + AF2 + AF3) packets, i = 1, 2, 3.

But in CRED,

AQS_{AFi} = TSW estimate based on the AFi packets alone.
where TSW refers to time sliding window technique of [8].

This is done in order to adhere to the AF PHB specifications [9], which state that the servicing of one AF class must be independent of the other AF classes.

5 Performance Evaluation

The performance of the proposed combination of the adaptive HPD and colored RED schemes is evaluated through simulation experiments. All experiments are performed using the network simulator ns-2 [10]. The robustness of the proposed scheme is tested using bursty (Pareto) traffic sources. The average value of the burst and idle times are set to 500 msec each and the distribution's shape parameter α is set to 1.2. All the TCP agents use the selective acknowledgment (SACK) mechanism. All packets are 1000 bytes in length.

The proposed scheme is also compared with a combination of the original HPD packet scheduler [1] and the RIO dropping scheme. For this combination, the same RIO parameters are used for all classes: (10, 20, 0.04) for OUT packets and (20, 40, 0.04) for IN packets, where the three parameters represent ($min_{th}, max_{th}, max_p$), respectively.

5.1 Simulation Setup

The topology (Fig. 2) used in all simulation experiments, consists of 9 sources and 9 destinations connected through 2 edge routers and 2 core routers. The edge routers have

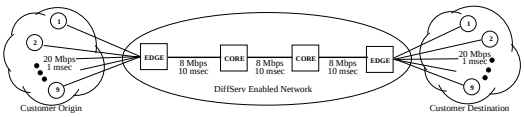

Figure 2. Simulation Topology

built-in packet meters, policers, and markers. The core routers only employ the proposed dropping algorithm and the proportional delay scheduler. The packet meters use the TSW technique [8] to compute each flow's instantaneous sending rate, based on which the packet's drop probability is computed. Packets from the customer network are classified (marked) to one of the 3 AF (green, yellow, or red) classes based on the service agreement. Further, the edge routers meter the flows and subject the packets to one of the two drop threshold levels. Table 2 gives the CRED parameter set used in the simulation experiments. In all the routers the buffer length is set high enough so that the queues never experience buffer overflows. In all the cases, the bottleneck link bandwidth is set to 8 Mbps. All the simulation experiments are performed for a period of 300 seconds.

Table 2. Parameters for CRED

	min_{th}	max_{th}	max_p
Red / AF11	20	40	0.08
Red / AF12	10	20	0.16
Yellow / AF21	20	40	0.04
Yellow / AF22	10	20	0.08
Green / AF31	20	40	0.04
Green / AF32	10	20	0.02

5.2 Results

Results are presented in the form of bar plots. The three horizontal lines in the throughput differentiation bar plot represent the target rates of the three AF classes: 380 Kbps, 760 Kbps and 1520 Kbps respectively. Likewise, the horizontal line in the delay differentiation bar plots represents the minimum one way end-to-end delay of 32 msecs that each packet would experience. Assuming negligible transmission, processing, and queuing delays, the 32 msec then consists only of the propagation delay.

Three different simulation experiments were performed with On-/Off- Pareto sources and constant bit-rate (CBR) sources. Firstly, packets from Pareto sources are carried over TCP SACK. Secondly, all traffic is carried over UDP. Lastly, one flow in each class carries CBR traffic over UDP, while the other flows carry Pareto traffic over TCP. The last experiment helps study the TCP/UDP interactions. Fig. 3, 4, and 6 show the corresponding results. It is obvious from the figures and the delay comparison Table 3 that a bandwidth, delay, and loss differentiation is achieved simultaneously in all the cases. In all the three experiments,

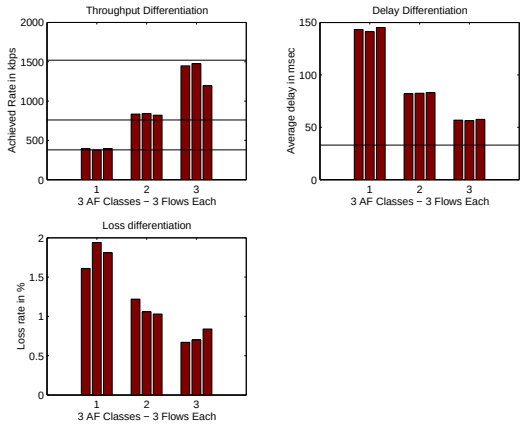

Figure 3. AHPD: Pareto over SACK

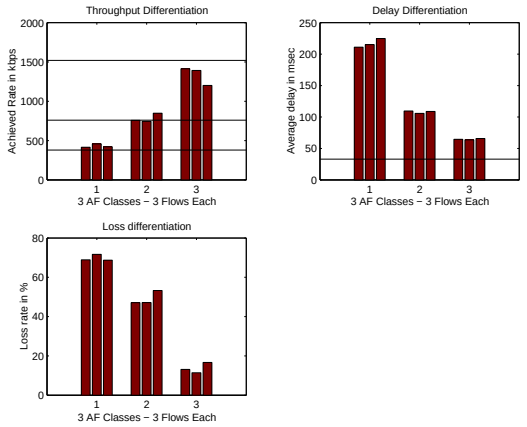

Figure 4. AHPD: Pareto over UDP

our scheme maintains the delay differentiation ratio better than the original HPD scheme.

In the first experiment (Fig. 3), one of the AF3 flows in Fig. 3 appears to receive a lesser share of the bandwidth. But, the fairness index[3] calculated proves otherwise. The fairness indices calculated for the 3 classes in the first experiment are 0.99, 0.99 and 0.99 for AF1, AF2 and AF3 classes respectively. It is obvious from Fig. 5 and 7 that the original HPD and RIO combination fail to bandwidth differentiate in the presence of UDP flows. On the other hand, our CRED scheme proportionally distributes bandwidth in the second experiment. In the third experiment, although bandwidth differentiation was achieved, the UDP flows get a slightly higher share of the bandwidth. The fairness indices calculated for this experiment are: 0.91, 0.91, and 0.99 for the 3 AF classes. This indicates that the TCP flows have got their fair share of bandwidth. With the increasing number of applications using UDP, the ability of our scheme to effectively bandwidth and delay differentiate

[3]Fairness Index = $\left[\sum_i x_i\right]^2 / [N . \sum {x_i}^2]$ where x_i is the mean throughput of traffic source i, and N is the total number of sources under consideration.

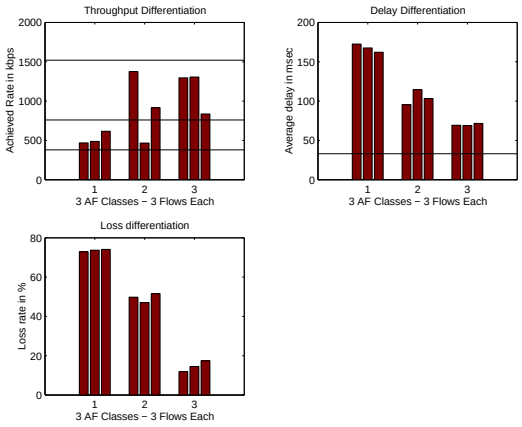

Figure 5. Original HPD: Pareto over UDP

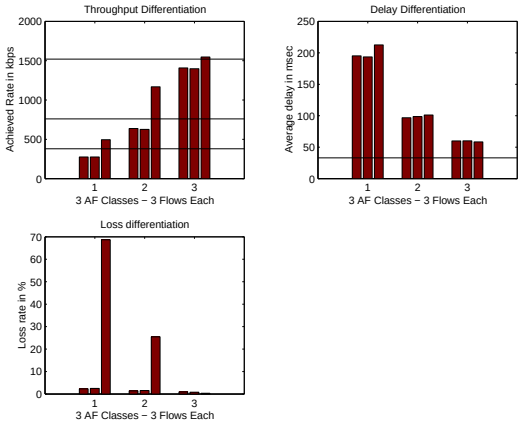

Figure 6. AHPD: CBR/UDP and Pareto/TCP

bursty traffic even in the presence of UDP flows supports our claim that our scheme is more robust. In all the three experiment loss differentiation is achieved. The UDP flows experience very high loss rates because, they do not regulate themselves when congestion takes place. In the third experiment, our dropping scheme strictly drops all the misbehaving UDP packets. These packets lost are in excess of their service agreement.

6 Conclusions

The need for maintaining the delay ratio between classes grows as the number of applications increases. Experimental results prove that the delay differentiation ratios obtained using our scheme is very much close to the ideal desired value. Moreover, our scheme also maintains the delay ratios better than the original HPD scheme. We achieve this by delaying the low priority class packets longer in the queue.

Loss rates and loss differentiation is better in the experiments which use congestion responsive transport agents (TCP) than those experiments which use non-

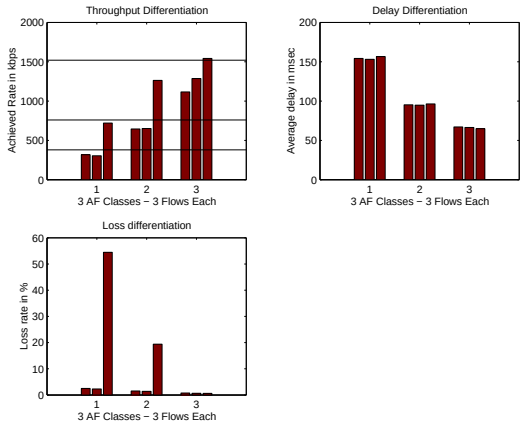

Figure 7. Original HPD: CBR/UDP and Pareto/TCP

Table 3. Delay Comparison

Traffic Type	Scheme	Average Class Delay msec			Delay Ratio	
		AF1	AF2	AF3	$\frac{AF1}{AF2}$	$\frac{AF2}{AF3}$
Pareto over TCP	AHPD & CRED	143.33	82.59	56.93	1.73	1.45
	HPD & RIO	135.96	86.39	61.40	1.57	1.41
Pareto over UDP	AHPD & CRED	181.27	97.22	63.12	1.86	1.54
	HPD & RIO	167.46	104.46	69.96	1.60	1.49
1 CBR/UDP & 2 Pareto/TCP	AHPD & CRED	200.51	98.85	59.49	2.03	1.66
	HPD & RIO	154.71	95.54	66.17	1.62	1.44

responsive transport agents (UDP). A good example of this case is the experiment with the CBR sources. Nevertheless, the non-responsive flows are also proportionally loss differentiated, but have to pay a price in the form of a higher loss rate. Also, CBR traffic, especially over UDP affects the fair distribution of resources in an uncontrolled environment. Our scheme effectively punishes the misbehaving UDP flows appropriately and distributes bandwidth proportionally. Since the scheduler works entirely based on the packet's class rather than the underlying transport mechanism, the delay differentiation is not affected due to the presence of UDP. For example, in the cases using UDP, the existing algorithm (HPD and RIO) even failed to bandwidth differentiate. While our scheme bandwidth differentiated in a far better manner.

In this paper, we have proposed a method to simultaneously control the bandwidth, delay, and packet loss in a proportional manner. We argue that the bandwidth, delay, and loss can be controlled simultaneously by acting on the delay and packet loss alone. Simulation results validate our claim. Comparison with other schemes shows our scheme's superiority.

References

[1] C. Dovrolis, "Proportional differentiated services: Delay differentiation and packet scheduling,"

IEEE/ACM Transactions on Networking, vol. 10, pp. 12–26, February 2002.

[2] M. K. H. Leung, J. C. S. Lui, and D. K. Y. Yau, "Adaptive proportional-delay differentiated services: Characterization and performance evaluation," *IEEE/ACM Transactions on Networking*, vol. 9, pp. 801–817, December 2001.

[3] L. Essafi, G. Bolch, and A. Andres, "An adaptive waiting time priority scheduler for the proportional differentiation model," in *Proc. ASTC HPC, Seattle, 2001*, April 2001.

[4] M. E. Markaki, M. P. Saltouros, and I. S. Venieris, "Proportional packet loss differentiation and buffer management for differentiated services in the internet," in *Proc. of 25th Annual IEEE Conf. Local Computer Networks 2000*, 2000.

[5] C. C. Li, S. L. Tsao, M. C. Chen, Y. Sun, and Y. M. Huang, "Proportional delay differentiation service based on weighted fair queuing," in *Proc. 9th Int. Conf. Computer Communications and Networks, 2000*, 2000.

[6] T. Soetens, S. Cnodder, and O. Elloumi, "A relative bandwidth differentiated service for tcp micro-flows," in *Proc. First IEEE/ACM Int. Symp. on Cluster Computing and the Grid, 2001*, 2001.

[7] G. Ruzzo and N. Chiminelli, "Wred tuning for bottleneck link," [Online] Available: http://carmen.cselt.it/papers/wred-cern/home.html.

[8] D. Clark and W. Fang, "Explicit allocation of best-effort packet delivery service," *IEEE/ACM Transactions on Networking*, vol. 6, no. 4, pp. 362–373, August 1998.

[9] J. Heinanen, "Assured forwarding phb group rfc 2597," June 1999.

[10] UCB/LBNL/VINT. (2002) The network simulator ns-2. [Online]. Available: http://www.isi.edu/nsnam/ns/

[11] C. Dovrolis and P. Ramanathan, "Proportional differentiated services, part ii: Loss rate differentiation and packet dropping," in *IEEE/IFIP Int. Workshop Quality of Service (IWQoS)*, June 2000.