Simultaneous and Proportional Bandwith, Delay, and Loss Differentiation [*]

Manimaran Selvaraj, Georgios Y. Lazarou and Rose Hu
Electrical and Computer Engineering
Mississippi State University
glaz@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. Traffic types used to study the performance of the scheme ranges from ordinary FTP data to bursty Pareto traffic. Numerical results validate our claim that regardless of the network traffic characteristics, especially burstiness, the adaptive delay scheduler combined with the packet dropper can effectively differentiate services in terms of delay, bandwidth and loss simultaneously. In addition, our proposed scheme showed on an average 11% improvement in performance when compared to the scheme proposed in [1].

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 issues like 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 applications can be differentiated in terms of one or a combination of 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 a service agreement. Thus, bandwidth differentiation is needed in this case.

Several scheduling algorithms [1]-[4] 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.

[*] Parts of this paper were presented at CIIT'03, ©IASTED 2003.

Many of these schemes were not robust to handle traffic under various network conditions. Moreover, complex off-line computations [3] or computationally demanding algorithms [2] were needed for some of the schemes.

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[1] packet dropper. Our adaptive scheduler is very robust and maintains an exact proportional delay differentiation under various network conditions. At the routers, our class based packet dropper works in tandem with our scheduler and achieves a relative bandwidth differentiation. The basis for a relative bandwidth differentiation is that a combination of the packet delay and loss of a flow reflects on the overall throughput achieved by the flow. Hence, 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:

$$\frac{\overline{d}_i}{\overline{d}_j} = \frac{\delta_i}{\delta_j} \qquad (i, j = 1...N) \tag{1}$$

The class weights $\{\delta_i\}$, are ordered as $\delta_1 > \delta_2 > ... > \delta_N > 0$, so 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 [2], and $\tilde{d_i}(t)$ is the normalized delay as calculated by PAD [3]. 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 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 the

[1]Random early detection [5].
[2]Average of waiting times of first packet in queue
[3]Average of delays experienced by all dequed packets

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

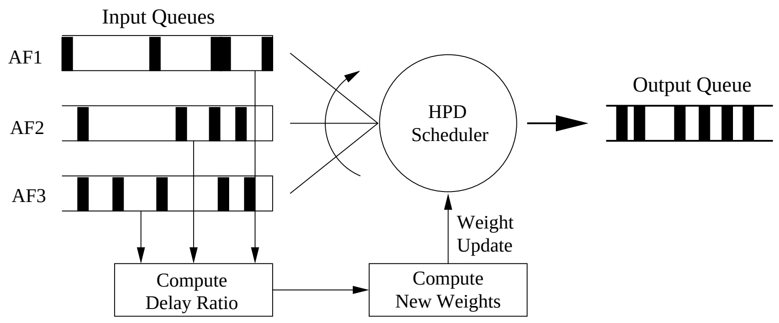

Figure 1: Adaptive HPD

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 Assured Forwarding (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. But this does not happen always. In our scheme, when the delay ratio is greater/lesser than K, the weights are adjusted, so that the delay ratio equals K. In order to avoid intense computational overhead, we relaxed the condition as follows. If the delay ratio falls inside a window $[K-\epsilon, K+\epsilon]$ 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 $[K - \epsilon,\ K + \epsilon]$. The weights are changed according to the weight function:

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

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 $\epsilon > 1$), would result in a performance similar to the original HPD scheme. In the weight function (4), Ψ and Φ are calculated as:

$$\Psi = \frac{(q^i_{max} - q^i_{curr}) \times |(K-\epsilon) - \Delta_i|}{(q^i_{max} - q^i_{min})}$$

$$\Phi = \frac{(q^i_{max} - q^i_{curr}) \times |\Delta_i - (K+\epsilon)|}{(q^i_{max} - q^i_{min})}$$

where q^i_{max} and q^i_{min} are the maximum and minimum possible values of a class weight respectively, and q^i_{curr} is the current value of the weight in a cycle. $|\Delta_i - (K \pm \epsilon)|$ represents the deviation of the computed delay differentiation ratio Δ_i from the window ($K-\epsilon, K+\epsilon$). $(q^i_{max} - q^i_{min})$ is the range of weights and $(q^i_{max} - q^i_{curr})$ 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 weight values are set as q_1 = 1, q_2 = 2, and q_3 = 4, ideally, 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.

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

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

1. Initialization. Set the initial parameters q_i, g, and the desired delay differentiation ratio Δ_i. Compute initial end to end average delay $\tilde{h}_i$. 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 average delay by using equation (2).

3. Select the queue with the greatest 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. The parameters are fixed according to the following relation:

$$\frac{RED\ drop\ probability\ of\ class\ i}{RED\ drop\ probability\ of\ class\ j} = \frac{\sigma_i}{\sigma_j} \tag{5}$$

where σ is the loss differentiation parameter. 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 identically. 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 the different 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 [11]. Several traffic types are used to evaluate the robustness of the proposed schemes. Notable sources are the On-/Off- traffic sources with burst and idle times taken from the Pareto and exponential distributions. The average value of the burst and idle times are set to 500 msec each. TCP and UDP flows are used in some of the experiments to study their interactions. All the TCP agents use the selective acknowledgment (SACK) mechanism. A fixed packet size of 1000 bytes is used in all the experiments. The value of 'g' in (2) is set at 0.85.

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

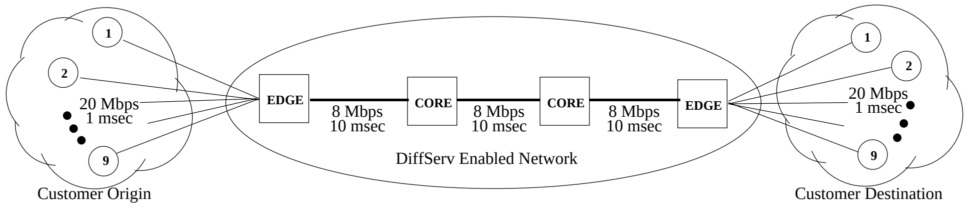

Figure 2: Simulation Topology

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 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 Time Sliding Window (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

Table 2: Run-Time Simulation Parameters of 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

are performed for a period of 300 seconds. The flows are preallocated bandwidth according to a service level agreement. Experiments were carried out using different traffic types so as to validate our claim that our scheme provides a proportional bandwidth, delay and loss differentiation under several network conditions.

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.

5.3 Experiments with FTP traffic over TCP SACK

In the first set of experiments, FTP traffic from greedy sources is carried over TCP by all the nine flows. Figure 3 shows the bandwidth, delay, and loss differentiation achieved. Figure 4 shows the differentiation achieved using the original HPD scheme. There is not much difference in the bandwidth differentiation between our scheme and the original HPD. But in the case of a delay differentiation, our scheme maintains the required delay differentiation ratios better than the original HPD scheme. In our scheme, the delay differentiation ratios Δ_1 and Δ_2 are very close to 2 (the desired value). This is achieved by the reduction in the delay levels of AF3 class and an increase in the delay levels of AF1 class. Table 3 shows the average delays and the delay ratios for each of the schemes. It is clear that our proposed scheme works better in maintaining the delay ratios between classes. It should also be noted that our scheme achieves lower loss rates than the original HPD scheme. Hence, traffic is differentiated

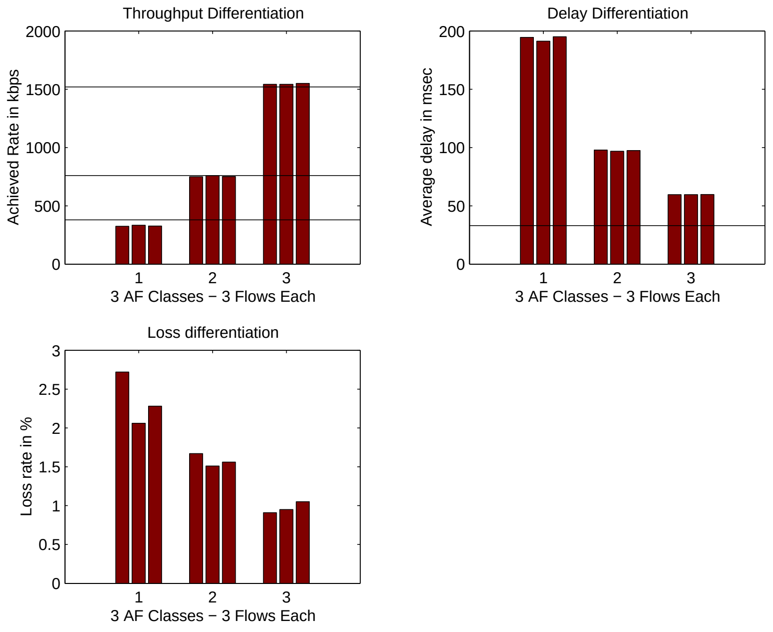

Figure 3: AHPD and CRED. FTP over TCP SACK. Bottleneck = 8 Mbps

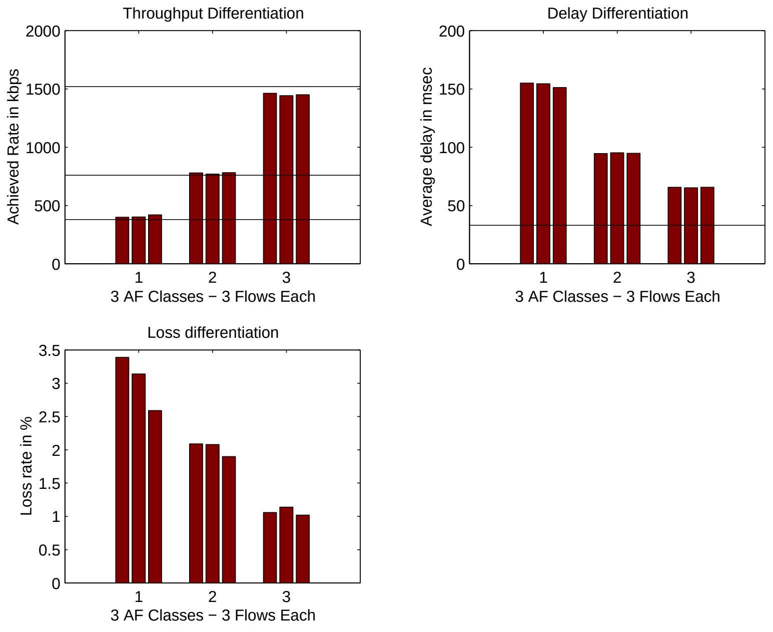

Figure 4: Original HPD and RIO. FTP over TCP SACK. Bottleneck = 8 Mbps.

Table 3: Delay Comparison for FTP Sources

Scheme	**Average Class Delay msec**			**Delay Ratio**	
	AF1	AF2	AF3	$\frac{AF1}{AF2}$	$\frac{AF2}{AF3}$
AHPD & CRED	193.67	97.4	59.66	1.99	1.63
HPD & RIO	153.6	94.94	65.53	1.62	1.45

well in terms of all three performance metrics.

5.4 Experiments with Constant Bit Rate Sources

Three different experiments are performed with CBR sources. In the first case, all the flows carry CBR traffic over TCP. In the second case, one flow in each AF class carries CBR traffic over UDP while the other flows carry FTP traffic over TCP. In the last experiment, all the flows consist of CBR traffic over UDP. Figures 5, 6 and 7 show the corresponding bandwidth, delay and loss differentiation for the 3 simulation experiments using our scheme. In all the cases, the CBR sources generate packets at a rate greater than their target rates. We see that our proposed scheme achieves the desired bandwidth, delay and loss differentiation in the first and third experiment. Bandwidth and delay differentiation is also attained in the second case, but the TCP/UDP interaction effect arises in this case. From Fig. 6, the UDP flows within each class appears to get a higher share of bandwidth than TCP. The fairness index[4] between flows within each class in the second experiment was calculated as 0.96, 0.97, 0.99 for AF1, AF2 and AF3 classes respectively. This level of fairness is quite acceptable for practical purposes.

A very important fact to note is that, the non responsive UDP flows are heavily punished when they try to exceed their allocated share of the bandwidth. They however still attain their target rates. All the packets lost by the UDP flows were in excess of their service level agreements. The severity of the UDP packet drop also depends on the packet's class. Hence, we can observe from Figure 6 that a loss differentiation is also achieved. Since TCP sources regulate themselves during congestion, they experience a far lower packet loss than UDP sources. Table 4 shows the delay comparison between the proposed scheme and the original HPD scheme. As in the earlier case with FTP, delay differentiation ratio is better maintained using our scheme than the original HPD scheme.

5.5 Experiments with Exponential Traffic Sources

Two different simulation experiments were performed with the On-/Off- traffic sources whose burst and idle times are exponentially distributed. In the first case, all the flows carry traffic from Exponential source over TCP and in the other case, all traffic is carried over UDP. Figures 8 and 9 show the corresponding results. In both the cases, a bandwidth, delay and loss differentiation is achieved. As in the earlier cases, the new scheme maintains the delay ratio better than the original HPD scheme. The delay comparison is shown in Table 5. In the experiment with traffic over UDP, the UDP flows experience losses greater than 50%. Thus the dropping scheme effectively punishes AF1 and AF2 classes when they

[4]The fairness index was calculated using the formula given in Appendix A.

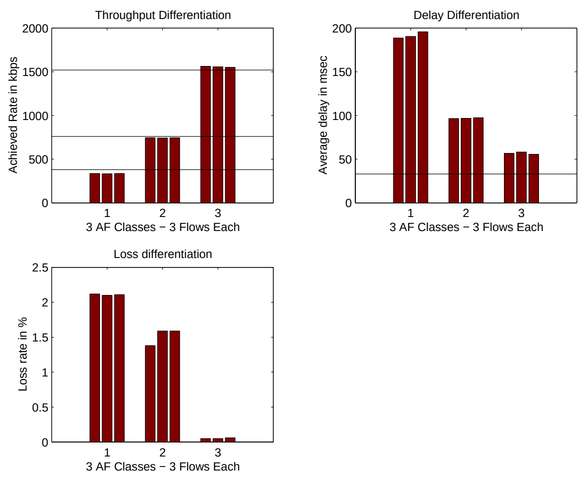

Figure 5: AHPD and CRED. All 9 flows are CBR over SACK

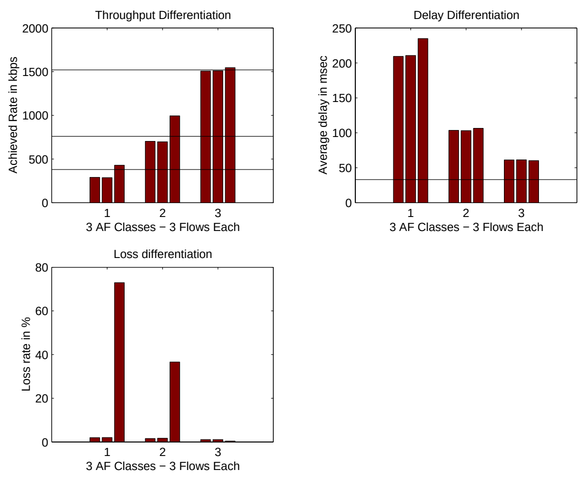

Figure 6: AHPD and CRED. Experiment with CBR. One flows in each class is CBR over UDP. Rest FTP/SACK

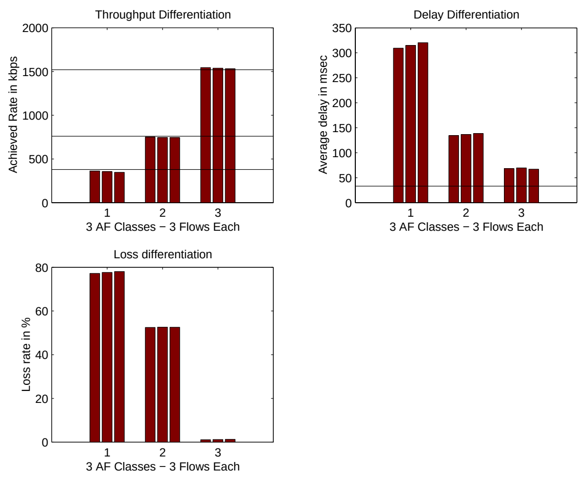

Figure 7: AHPD and CRED. All 9 flows are CBR/UDP

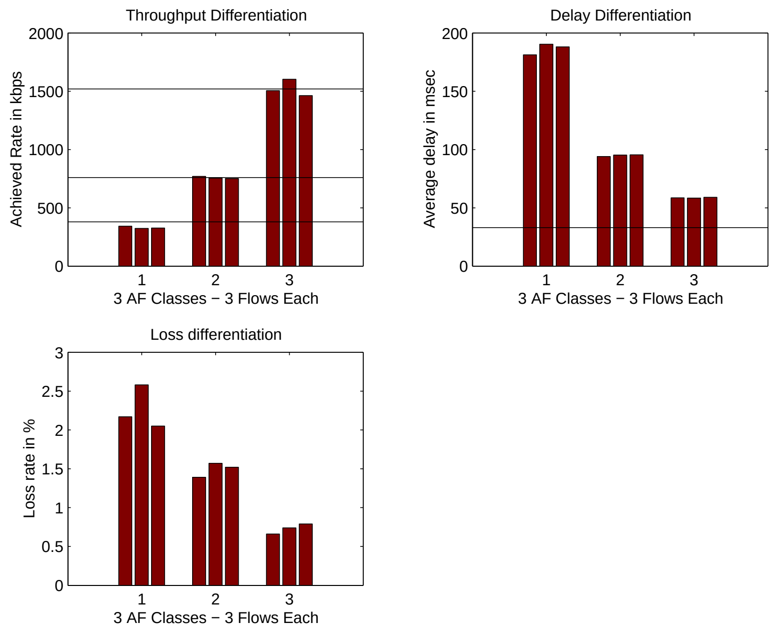

Figure 8: AHPD and CRED. Exponential traffic over SACK

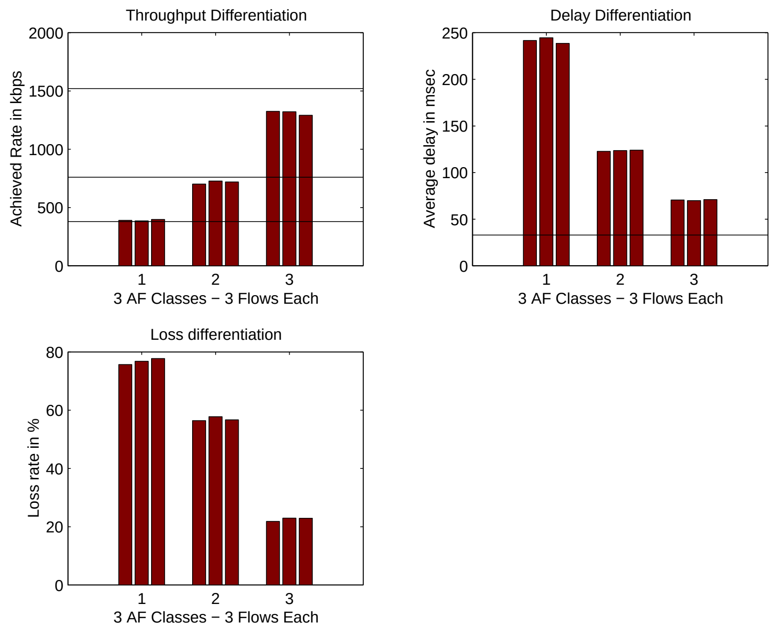

Figure 9: AHPD and CRED. Exponential traffic over UDP

Table 4: Delay Comparison for CBR Sources

Traffic Type	**Scheme**	**Average Class Delay msec**			**Delay Ratio**	
		AF1	AF2	AF3	$\frac{AF1}{AF2}$	$\frac{AF2}{AF3}$
All flows are CBR/ TCP	AHPD & CRED	191.65	96.90	56.83	1.97	1.7
	HPD & RIO	152.55	94.47	65.36	1.61	1.45
One flow in each class CBR/ UDP	AHPD & CRED	218.33	104.43	60.92	2.09	1.71
	HPD & RIO	170.48	103.24	69.67	1.65	1.48
All flows are CBR/ UDP	AHPD & CRED	314.78	136.53	68.29	2.3	1.99
	HPD & RIO	216.49	126.32	81.13	1.71	1.56

try to obtain a greater share of the link capacity. Again our scheme maintains the delay differentiation ratio better than the original scheme.

Table 5: Delay Comparison for EXP Sources

Traffic Type	**Scheme**	**Average Class Delay msec**			**Delay Ratio**	
		AF1	AF2	AF3	$\frac{AF1}{AF2}$	$\frac{AF2}{AF3}$
Exponential over TCP	AHPD & CRED	186.64	94.94	58.66	1.97	1.61
	HPD & RIO	151.83	94.07	65.10	1.61	1.45
Exponential over UDP	AHPD & CRED	241.61	123.46	70.49	1.96	1.75
	HPD & RIO	217.43	128.93	82.38	1.68	1.56

5.6 Experiments with Pareto Traffic Sources

Three different simulation experiments were performed with On-/Off- Pareto sources and constant bit-rate (CBR) 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. 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. Figures 10, 11, and 13 show the corresponding results. It is obvious from the figures and the delay comparison Table 6 that a bandwidth, delay, and loss differentiation is achieved simultaneously in all the cases. In all the three experiments, our scheme maintains the delay differentiation ratio better than the original HPD scheme.

In the first experiment (Fig. 10), one of the AF3 flows appears to receive a lesser share of the

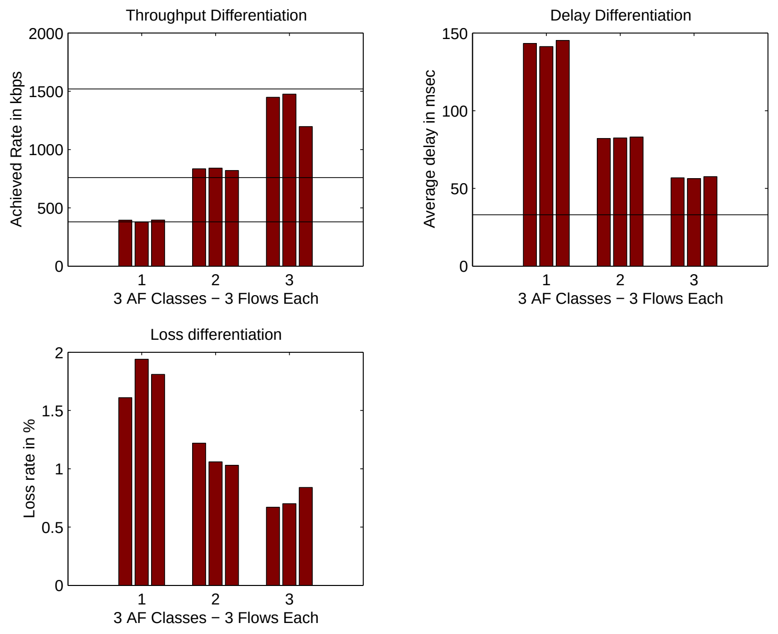

Figure 10: AHPD and CRED: Pareto over SACK

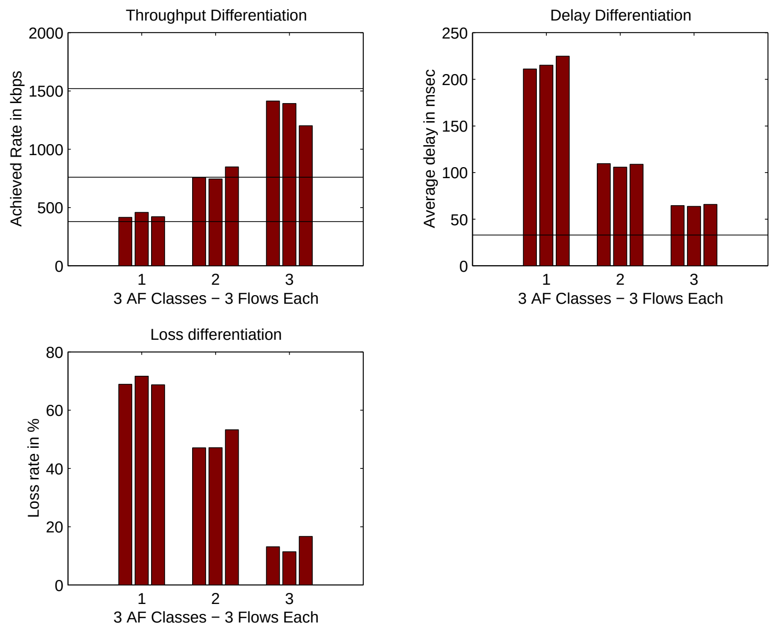

Figure 11: AHPD and CRED: Pareto over UDP

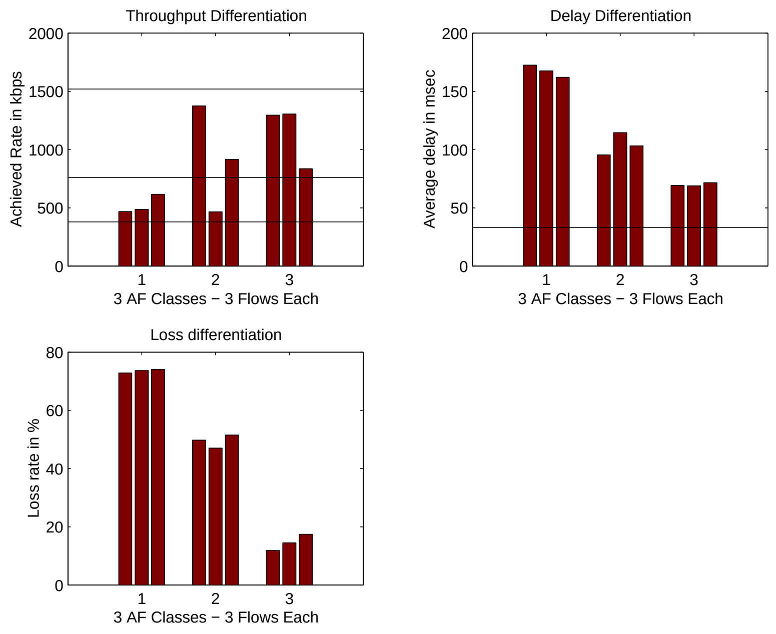

Figure 12: Original HPD: Pareto over UDP

bandwidth. But, the fairness index 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.

Table 6: Delay Comparison for Pareto Sources

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

It is obvious from Fig. 12 and 14 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 bursty traffic even in the presence of UDP

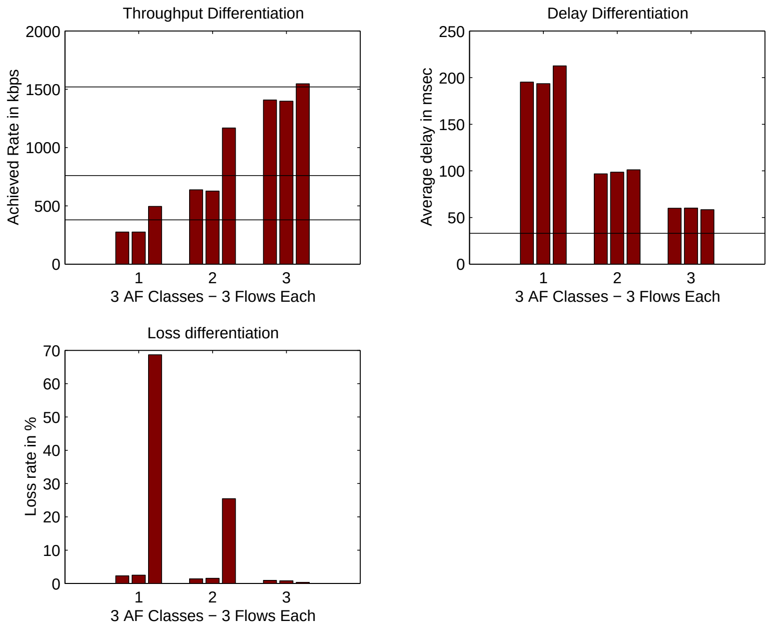

Figure 13: AHPD and CRED: CBR/UDP and Pareto/TCP

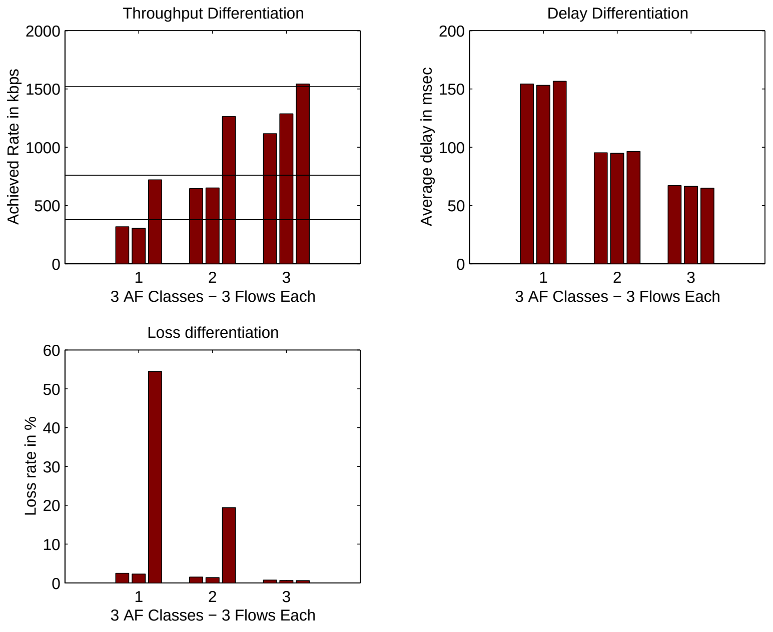

Figure 14: Original HPD: CBR/UDP and Pareto/TCP

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.

5.7 Experiments with Flows Differing in RTT

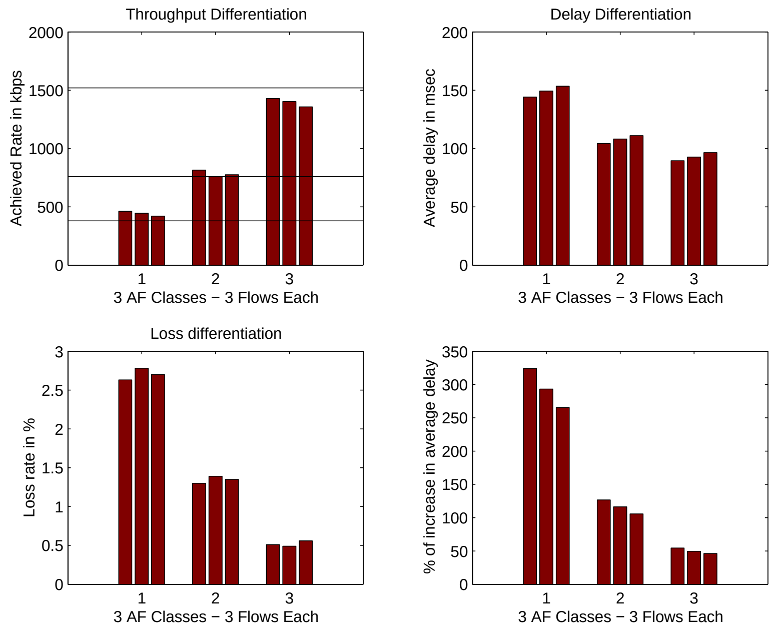

Figure 15: Original HPD: Flows with Different RTT

The performance of the proposed scheme is tested with flows having different round trip times (RTT). The nine flows starting from class AF1 to AF3 had RTTs of 68, 76, 84, 92, 100, 108, 116, 124, and 132 msecs respectively. Thus, a higher priority flow had a higher RTT than a lower priority flow. Figures 15 and 16 show the performance of the original HPD scheme and the proposed scheme respectively. The proposed CRED scheme suppresses the effect of difference in RTT between the flows and thus the classes get a proportional share of bandwidth. On the other hand, the bandwidth ratios between classes is affected in the experiments with the original scheme. This is because the low priority classes, due to their lower RTTs, get a greater than allocated share of bandwidth. Table 7 shows the average bandwidth

Table 7: Bandwidth Comparison for RTT Experiment

Scheme	**Average Class Bandwidth kbps**			**Bandwidth Ratio**	
	AF1	AF2	AF3	$\frac{AF2}{AF1}$	$\frac{AF3}{AF2}$
AHPD & CRED	415.12	761.11	1433.90	1.83	1.89
HPD & RIO	442.43	782.98	1397.22	1.77	1.78

obtained by the different classes and the bandwidth ratios between the AF classes for the two test cases.

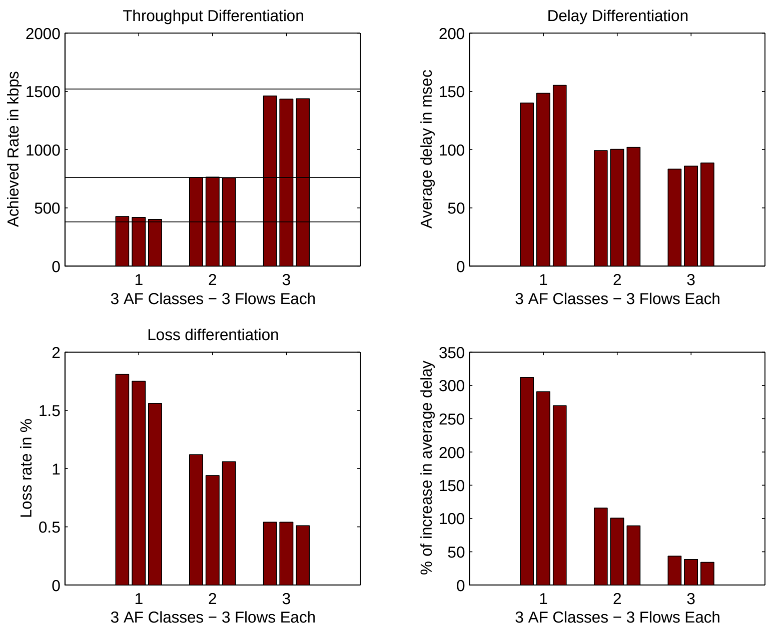

Figure 16: AHPD and CRED. Flows with Different RTT

Table 8: Delay Comparison for RTT Experiment

Scheme	**Average Class Delay msec**			**Delay Ratio**	
	AF1	AF2	AF3	$\frac{AF1}{AF2}$	$\frac{AF2}{AF3}$
AHPD & CRED	147.97	100.50	85.90	1.47	1.17
HPD & RIO	149.02	107.87	92.93	1.38	1.16

It is obvious that our proposed CRED scheme achieves a proportional bandwidth differentiation better than the RIO scheme.

Regardless of the low priority classes having lower RTTs, both the original HPD and our scheme succeed in achieving a delay differentiation. Table 8 shows the delay ratio comparison between our scheme and the original HPD scheme. The difference in the RTT has however affected the delay ratios between the classes. Again, our scheme shows an improved performance over the original HPD scheme. Figures 15 and 16 also show the percentage of the increase in the delay of the individual flows from their ideal one way delay values. Table 9 gives the corresponding numerical values. In most of the cases our scheme increases the end-to-end delay to a lesser extent than the original scheme. Hence, in spite of a difference in RTT, our scheme achieves proportional delay, loss and bandwidth differentiation simultaneously.

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

Table 9: Percentage of Delay increase from ideal values.

Adaptive HPD	**Original HPD**
312.00	323.97
290.71	293.11
269.69	265.55
115.80	126.80
100.62	116.32
88.93	105.79
43.53	54.44
38.48	49.52
34.26	46.22

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-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. The experiment with Pareto sources further augments our claim that our scheme is more robust. Our scheme out-performed the existing HPD and RIO combination in these tests with bursty traffic. For example, when bursty traffic was carried over UDP, the existing algorithm (HPD and RIO) even failed to bandwidth differentiate. While our scheme bandwidth differentiated in a far better manner.

In all experiments, our proposed scheme (AHPD and CRED) showed several degrees of improved performance when compared to the existing (HPD and RIO) scheme. The performance improvement factors are: 14% for FTP traffic, 17.95% for CBR traffic, 13.1% for Exponential traffic, and 9.6% for Pareto traffic. In the last experiment with flows varying in RTT, our scheme achieved a 3.3% improvement in performance. As mentioned earlier, this is due to the difference in RTTs of the flows. Thus, on an average, our scheme provides at least 11% improvement over the existing technique.

This work also gave birth to new ideas along the same directions. These ideas if successful would contribute greatly to proportional differentiated services model. In the HPD scheme, the factor 'g' decided the balance between WTP and PAD. The scheduling scheme can be made more robust by making this factor adaptive to packet arrival pattern. Hence, as the arrival pattern of packets varies over a small window of time, the factor 'g' can also be varied accordingly. In addition to this, the CRED packet dropper can be made more robust by maintaining a history of the packet loss rate. The count of packets lost by every class can be compared and their respective drop probabilities can be modified accordingly. This would help maintain the proportional loss differentiation in a robust manner. A much broader topic to work on from this stage would be to test the effectiveness of the proposed DiffServ techniques in a Multiprotocol Label Switching (MPLS) enabled network[11][23]. Several recent works have contributed greatly to the success of a Diff-Serv architecture over laid on a MPLS network. The proposed scheme can be further fine tuned by making use of MPLS's traffic engineering mechanisms.

To conclude, 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 and P. Ramanathan, "Proportional Differentiated Services: Delay Differentiation and Packet Scheduling," *IEEE/ACM Trans. on Networking.*, 10:12–26, February 2002.

[2] M. Leung, J. Lui, and D. Yau, "Adaptive Proportional Delay Differentiated Services: Characterization and Performance Evaluation," *IEEE Trans. on Networking.*, 9(6):801–817, December 2001.

[3] L. Essafi, G. Bolch, and A. Andres, "An Adaptive Waiting Time Priority Scheduler for the Proportional Differentiation Model," *Proc. of ASTC HPC.*, April 2001.

[4] M. Markaki, M. Saltouros, and I. Venieris, "Proportional Packet Loss Differentiation and Buffer Management for Differentiated Services in the Internet," *Proc. of the 25th Annual IEEE Conference on Local COmputer Networks*, pages 306-313, June 2000.

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

[6] T. Soetens, S. D. Cnodder, and O. Elloumi, "A Relative Bandwidth Differentiated Service for the TCP Micro-flows," *Proc. First IEEE/ACM Int. Symp. on Cluster Computing and the Grid.*, November 2001.

[7] G. Ruzzo and N. Chiminelli, "WRED Tuning for Bottleneck Link," [Online] Available: http://carmen.cselt.it/papers/wred-cern/home.html, February 2000.

[8] D. Clark and W. Fang, "Explicit Allocation of Best-effort Packet Delivery Service," *IEEE Trans. on Networking*, 6(4):362–373, August 1998.

[9] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured Forwarding PHB Group," *Request for Comment - RFC 2597.*, June 1999.

[10] R. Jain, *The Art of Computer Systems Performance Analysis*. John Wiley and Sons, 1991.

[11] UCB/LBNL/VINT. *Ns-2 Network Simulator*. [Online] Available: http://www.isi.edu/nsnam/ns/, 2002.

A Fairness Index

Fairness index was calculated using the following formula as given in [10]:

$$Fairness\ \ Index\ \ FI = \frac{[\sum_i x_i]^2}{N.\sum {x_i}^2}$$

where fairness index ranges between 0 and 1 and x_i is the mean throughput of traffic source i, and N is the total number of sources under consideration. The closer the fairness index to 1, the fairer is the bandwidth distribution between sources.