A Buffer-Management Scheme for Bandwidth and Delay Differentiation using a Virtual Scheduler

Roman Pletka[†,*], Patrick Droz[†] and Burkhard Stiller[‡]

[†]IBM Research, Zurich Research Laboratory
8803 Rüschlikon, Switzerland
{rap,dro}@zurich.ibm.com
[*]Corresponding author. Phone: +41-1-724-8351, Fax: +41-1-724-8954

[‡]Computer Engineering and Networks Laboratory (TIK)
ETH Zürich, 8092 Zürich, Switzerland
stiller@tik.ee.ethz.ch

Abstract—**This paper presents a new scalable buffer-management scheme for IP Differentiated Services. The scheme consists of a Differentiated Random Drop (DRD) algorithm using feedback from a virtual scheduler. DRD choses a queue to perform an early packet drop to avoid congestion according to a specific probability function. First it will be shown that DRD in conjunction with first-come first-served scheduling is able to support relative service differentiation. The virtual scheduler is introduced to enable service differentiation in terms of bandwidth and delay at the same time. A virtual scheduler runs in parallel to the real scheduler and maintains virtual queue lengths that are being used by the congestion avoidance scheme as a feedback for packet drop decisions. Scheduling packets for transmission is performed by the real scheduler only.**

Keywords— **Quality-of-Service, Differentiated Services, Relative Service Differentiation, Bandwidth and Delay Guarantees**

I Introduction

In the past few years many different scheduler and queue-management algorithms have been proposed. Research activities have been and still are focused on how to satisfy the Quality-of-Service (QoS) requirements of higher-priority flows while keeping fairness among classes and preventing starvation of low-priority traffic.

In Weighted Fair Queueing (WFQ) schedulers such as Self-Clocked Fair Queueing (SCFQ) [1] and other rate-proportional service disciplines [2], [3], [4], queue weights are used to provide per-flow[1] bandwidth guarantees: The link share of backlogged connections is proportional to queue's weight, and excess bandwidth is distributed in the same manner. Flows that are significantly below their reserved bandwidth share will experience less delay. This means that delay-sensitive services such as Voice-over-IP (VoIP) can be implemented using WFQ only in combination with a token bucket at the ingress node and that the token bucket rate is set lower than the reserved rate. In Class-Based Queueing (CBQ) [5], delay differentiation can be achieved using a priority-based packet scheduling algorithm for bounded traffic classes. In all these different schemes, thresholds (for scheduler and token buckets) have to be set carefully and often their meaning is not intuitive. Usually, this is done in a static manner when the network is set up and often default parameters are not modified at all.

This paper focuses on a new threshold-based buffer-management scheme that consists of a combination of Differentiated Random Drop (DRD) [6] and virtual scheduling. Unlike other known schemes, the proposed scheme supports simultaneous bandwidth and delay differentiation and has the following advantages:

- Dynamic drop rate adaption of traffic classes,
- efficient and early congestion avoidance, and
- easy setting up of thresholds.

It will be shown that the scheme is suitable for a Diffserv [7] enabled network, where it can be used to implement relative QoS guarantees in Assured Forwarding (AF) [8] per-hop-behaviors.

The elements of active buffer management are algorithmic droppers, packet-marking strategies, and scheduling algorithms. Several extensions that are combined in a flow-and-queue threshold-based buffer-management scheme use active buffer-management elements. These extensions are fundamental to fair packet dropping and better overall buffer usage. Furthermore DRD is briefly introduced as an efficient congestion avoidance algorithm. It will be shown that DRD in conjunction with threshold-based buffer management and simple first-come first-served (FCFS) scheduling is able to provide service differentiation in terms of dynamic and adaptive packet drop rates, which are relative to other queues in the system. The efficiency of a virtual scheduler, which is the key to bandwidth and delay differentiation, is compared to that of a simple FCFS scheduler using extensive simulations real-

[1]In general the term micro-flow is defined by the 5-tuple of source and destination address and port number, and the protocol number in the IP header of the packet. Macro-flows may consist of a large number of micro-flows that form a flow aggregate. For the sake of simplicity, this paper refers to flow aggregates simply as flows, and to a single micro-flow as a flow hence.

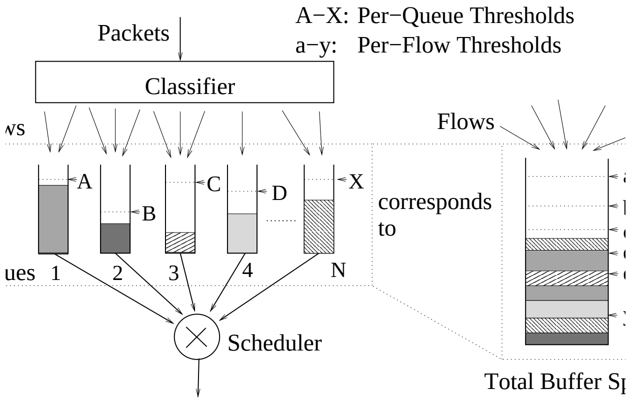

Fig. 1. *Architecture of buffer management.*

ized in a modified version of the network simulator *ns* [9] that was specifically extended for this purpose.

The remainder of this paper is organized as follows. Based on the detailed discussion of the threshold-based buffer management in Section III, the virtual scheduler is introduced in Section IV. A simulation model utilizes the Diffserv Architecture to provide insights of the combination of FCFS and DRD with a virtual scheduler (Section V). Finally, Section VI summarizes the advantages and draws conclusions.

II Related Work

In [10] Drovolis et al propose a proportional differentiation model to refine and quantify relative service differentiation. Two packet schedulers that approximate the model are introduced and evaluated in simulations. The proportional model is applied on queueing-delay differentiation only and leaves the problem of coupled delay and loss differentiation for future work.

In [11] Risso discusses Decoupled Class Based Scheduling (D-CBQ), a CBQ-derived scheduling algorithm that uses new link-sharing guidelines to decouple bandwidth and delay for bounded classes. The algorithm improves

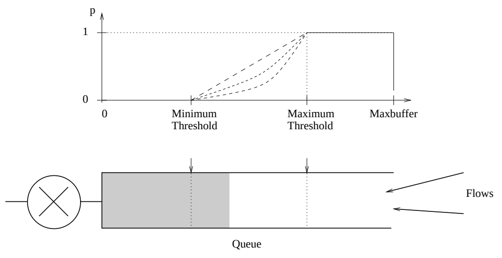

Fig. 2. *Examples of probability functions.*

delay characteristics of bounded classes compared to CBQ. Whereas setting higher priorities (that means lower delays) no longer leads to more bandwidth allocated, incoming traffic still has to be limited by means of an additional token bucket filter. Note that the impact on delay and bandwidth using unbounded classes has not been studied in a severely overcharged network environment.

III Threshold-based buffer management

Figure 1 shows a flow-and-queue threshold-based buffer-management scheme [6]. Thresholds are assigned to flows and queues. Flows may consist of a large number of micro-flows that form a flow aggregate. As can be seen on the left-hand side, each of these flows is attributed to one queue and several flows can enter the same queue. In general, packets with the same QoS needs will enter the same queue although there may be multiple queues having approximately the same properties to differentiate for example between TCP and UDP traffic. On the right-hand side, the same flows are shown in the context of overall buffer space. The process of packet classification will not be discussed as it would exceed the scope of this paper.

As indicated by its name, flow-and-queue threshold-based buffer management is a scheme primarily based on two thresholds. The first threshold limits global buffer occupancy of a flow and is called the **per-flow threshold**. This means that flows exceeding their per-flow threshold undergo a special treatment such as marking or dropping packets. Marking and dropping depends on the type of buffer management and will be discussed later. Per-flow thresholds are measured relative to the total used buffer space.

The second threshold is a **per-queue threshold**, which allows a segmentation of the available buffer space and is compared to the buffer space used by this queue. When the per-queue threshold is exceeded, packets have to be dropped to limit the maximum packet delay. When used with no additional strategies, the per-queue threshold acts as a “hard” dropping policy. Hard means that packets may be suddenly dropped in bursts when the queue size exceeds the threshold. Clearly this behavior is not at all desirable. An early dropping policy such as Random Early Detection (RED) [12] or DRD should be combined with this thresh-

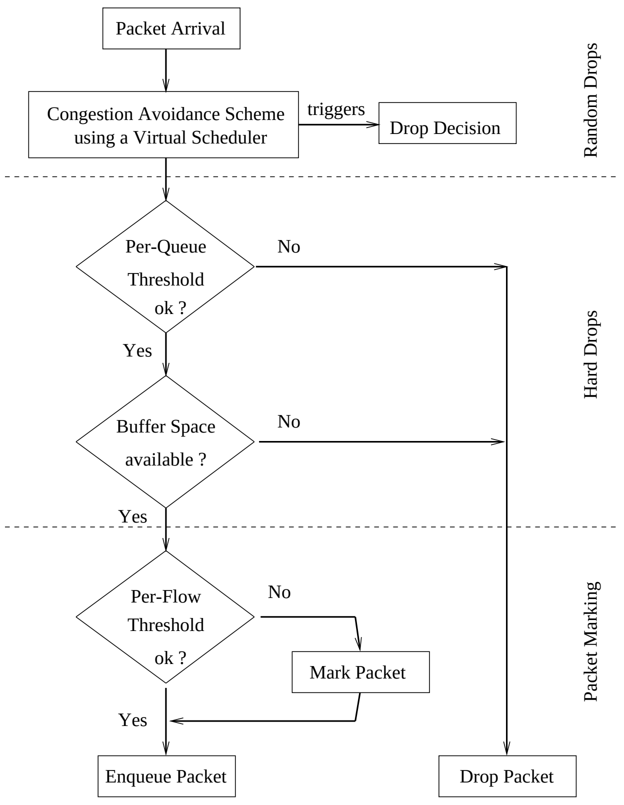

Fig. 3. *Algorithmic dropper.*

old. For this purpose DRD will be introduced in the next Section. Later in this paper it will be shown that DRD outperforms traditional RED and has additional useful properties.

Using per-queue thresholds allows more than the real existing buffer space to be allocated to queues as opposed to hard segmented buffer spaces as used in [13]. This means that the sum of all per-queue thresholds may exceed the total available buffer space. The advantage of such a strategy is that it supports larger bursts of a flow when other flows are on a low buffer-usage level or not backlogged at all and, therefore, uses less global buffer space. On the other hand when all flows send at a peak rate, the fixed per-queue buffers cannot be fully exploited at the same time. At this point the per-flow threshold will act as a limiter. In conjunction with RED or DRD, this limit is not a hard limit and therefore does not cause bursty packet drops. The service rate will no longer be absolute but rather relative to other classes. This is even mandatory for giving best-effort traffic the capability to take advantage of unused bandwidth.

III-A Hard dropping scheme

The simplest buffer-management scheme known consists of dropping packets when no more buffer space is available. This strategy commonly used in the past and even today, turns out to be inadequate for performing efficient and fair packet forwarding even when used with fair queuing. Packets are often dropped in “bursts” from a single flow, whereas other flows increase their traffic even more. As a result, fairness suffers and QoS requirements simply cannot be guaranteed.

Adding flow-and-queue threshold-based buffer management enables buffer sharing and priority handling. In addition to dropping when no buffer space is available, packets are dropped when one or both thresholds are exceeded. The simulations discussed in [6] show that such a simple flow-and-queue threshold-based buffer management is not sufficient per se to guarantee services as defined in Diffserv.

In general, the thresholds used in this mechanism act as a hard limit. During congestion periods no indication is performed, and packets are suddenly dropped in large bursts once a threshold has been exceeded. There must be some additional packet-dropping strategies to avoid burst drops and to avoid synchronization of TCP sources.

III-B Softening hard limits

One way to overcome the hard dropping nature of a threshold is to introduce a steadily increasing probability function depending on the average queue size (Figure 2). A linearly increasing function is used because of its simplicity: it is sufficient to add an additional threshold that can be as simple as a default percentage of the per-flow threshold. The two thresholds together are then used as a lower and upper limit (max/min per-queue thresholds).

The size of the linearly increasing section has to be set relative either to the global buffer space or to allocated queue space. Too small a value does not overcome the bursty drop problem and too large a size of the linearly increasing section will introduce premature and unnecessary packet drops. Without going into more details concerning the optimum setting, which would be beyond the scope of this paper, experiments have shown that a value of approximately 50% is reasonable [12].

III-C Introducing packet marking

Service differentiation within a flow can be achieved by introducing a set of drop precedences. A rather simplistic approach would be to just mark all packets that have a higher precedence than the low default drop precedence. In doing so, however, almost exlusively packets having a higher drop precedence will be dropped and differentiation between more than two drop precedences will no longer be feasible. Therefore, the proposition is to assign per-flow thresholds to drop precedences and each drop precedence gets its per-flow threshold. As mentioned, multiple flows with different per-flow thresholds may coexist within the same queue. Marked packets in a queue may belong to different flows and no distinction according to the initially

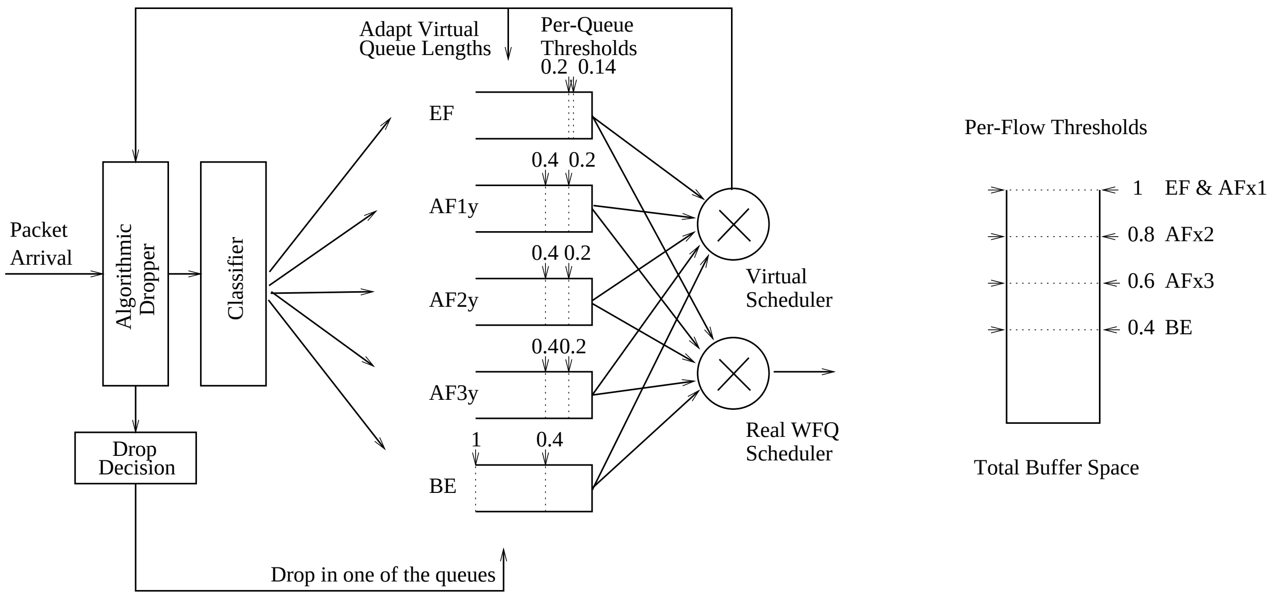

Fig. 4. *Architecture of buffer management.*

given drop precedence is done. The marking is used when drop decisions have to be made. Thus the per-flow drop rate increases with decreasing per-flow threshold. Packet marking can be seen as a previous conviction with a scope that is strictly limited to the actual router.

If per-flow thresholds are used to trigger packet drops, it is easy to see that a small per-flow threshold will prevent a flow completely from obtaining any service when network traffic is high. A part of the global buffer space remains unused at this time because it is reserved for other traffic classes that perhaps will not occupy this space in the near future. One way to improve buffer usage is to increase the per-flow threshold but then service differentiation becomes more difficult because these thresholds move closer together. Nevertheless, an arriving packet belonging to such a flow could be enqueued and marked. If packets need to be dropped later, those marked should be dropped first. As a result buffer space is used more efficiently and more packets are served overall. A sophisticated packet-dropping scheme can take into account the per-queue buffer usage as well as a relative queue priority, and then select a packet to be dropped.

III-D Algorithmic Dropper and Congestion Avoidance Scheme

Upon packet arrival, the algorithmic dropper examines wether the packet should be enqueued and an additional action taken. An additional action is an action that tries to prevent congestion in the network. Figure 3 illustrates the algorithmic dropper used in our scheme. The algorithmic dropper consists of three parts: The first is used for congestion avoidance and evaluates whether an existing packet has to be dropped in one of the queues by choosing a queue randomly. If yes, a packet drop in this queue is triggered. The second part consists of hard dropping limits (tail-drop) for queue and buffer overflows. The congestion avoidance scheme should drop packets earlier so that tail-drops occur only rarely. In the third part, packet marking to implement drop precedences in a queue occurs.

The congestion-avoidance block uses DRD to evaluate potential packet drops. The main goal of the DRD scheme is to introduce a dynamic per-queue drop probability while adding relative dependency among the various queues in the system. This will primarily allow service differentiation such as “better than” another class. Each time a packet arrives, the following congestion-avoidance mechanism is performed before processing of that packet continues. Using a dynamic per-queue probability, one of the queues is chosen randomly and random early discard is then performed in this queue. The per-queue probability p_i is evaluated as follows: Every queue is assigned a fixed priority equal to the queue number i. Thus the queues are sorted according to their priority. The per-queue probability is proportional to the number of bytes in the queue plus the number of bytes in all higher-priority queues. This is a more general approach than is used for RIO in [14]. Clearly queues containing no packets have zero per-queue probability. Note that priorities are introduced only for dropping behavior and not, as for example in CBQ [5], as a per-queue priority used for scheduling purposes. In addition, higher priority does not imply lower packet delay. The per-queue probability p_i can be written as

$$p_i = \begin{cases} C \sum_{k=1}^{i} b_k & \text{if } b_i \neq 0 \\ 0 & \text{if } b_i = 0 \end{cases}, \qquad (1)$$

where b_k is the number of bytes in queue k. For N queues

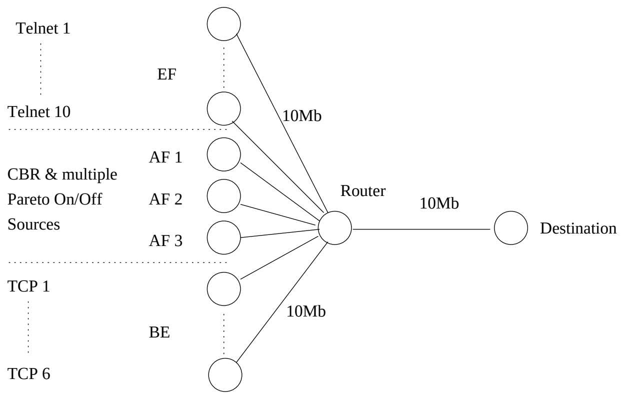

Fig. 5. *Simulation topology.*

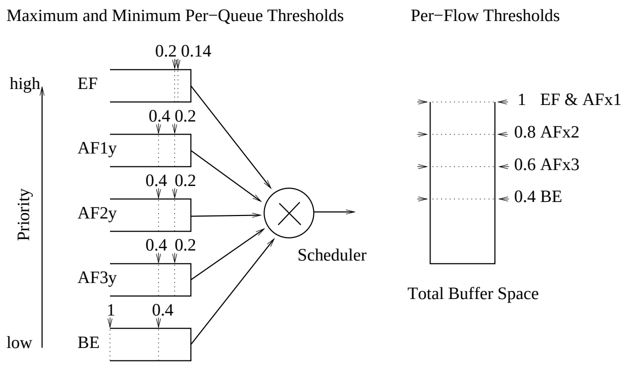

Fig. 6. *Per-flow and per-queue thresholds.*

the normalization is

$$\sum_{i=1}^{N} p_i = 1 \; , \tag{2}$$

and the constant C is then given by N queues

$$C = \frac{1}{\sum_{j=1, b_j \neq 0}^{N} \sum_{k=1}^{j} b_k} \; . \tag{3}$$

Finally, if a packet in that queue has to be dropped, by preference a marked one is chosen.

IV Virtual Scheduler

A scheme as the one illustrated in the preceding section is able to support relative service differentiation even with a simple FCFS scheduler [6]. Service classes in terms of “better than” can be implemented, and differentiation is expressed in lower drop rates for lower-numbered (higher-priority) queues. Whereas the scheme is able to support minimum bandwidth guarantees and fair excess bandwidth allocation, it fails in differentiating packet delays due to the simple FCFS scheduler. The idea is to combine two schedulers while keeping their advantages: The first scheduler will maintain fair packet scheduling and enable delay differentiation. For this a WFQ scheduler can be used. The second scheduler will be responsible for early congestion avoidance and start dropping packets if needed. It maintains virtual queue lengths used by the congestion avoidance scheme as a feedback. This scheduler is called virtual because it does not directly influence the departure time of packets in the buffer. Its result, the virtual queue lengths, are only used by the algorithmic dropper to perform drop decisions.

Figure 4 illustrates the architecture of the buffer management scheme using a virtual scheduler. As basis the scheme from Section III has been taken and enhanced to support a virtual scheduler. The buffer is divided into several queues. The number of queues is configurable, and the queues are served in a WFQ manner. Within each queue several parameters are given (queue number, max/min per-queue threshold and queue weight). In contrast to other schemes, these parameters are fixed at the beginning once and for all, and no tuning is required later on. In addition, the parameters in the queues with relative delay differentiation are equal (queues 2 to 4), thus making configuration easy. In Section V it will be shown that such a scheme is capable of providing a dedicated service class to a given queue. When a packet arrives, it will go through the algorithmic dropper with the only modification, that virtual queue lengths are used instead of the real ones, to trigger a packet drop. Meanwhile all packets are also served by the virtual FCFS scheduler.

Buffer management becomes quite difficult because the two schedulers serve packets simultaneously. Therefore, a special packet tag that is attributed to each of the packets and contains all necessary information has been introduced. In other words, if a packet has been treated by the real scheduler (and therefore has been sent to the outgoing link) but is not yet served by the virtual scheduler, the packet tag will remain stored in memory while the space used for the real packet can be freed. If a packet has been treated first by the virtual scheduler but not yet by the real scheduler, the packet tag and the packet itself will remain stored in memory until the packet has been served by the real scheduler. It is clear that by introducing packet tags, which can remain in memory longer than a packet lifetime, the overall memory usage will increase. Section V discusses this issue, and shows that memory increase is limited.

IV-A *Parameter setting guidelines*

The following guidelines should help set the parameters of a flow-and-queue threshold-based buffer-management scheme with N queues:

- The queue weight w_i corresponds to the minimum bandwidth guarantee for the service class in queue i and is a part of the Service Level Agreement (SLA).
- For equal per-queue threshold settings, lower-delay classes are in higher-numbered queues.

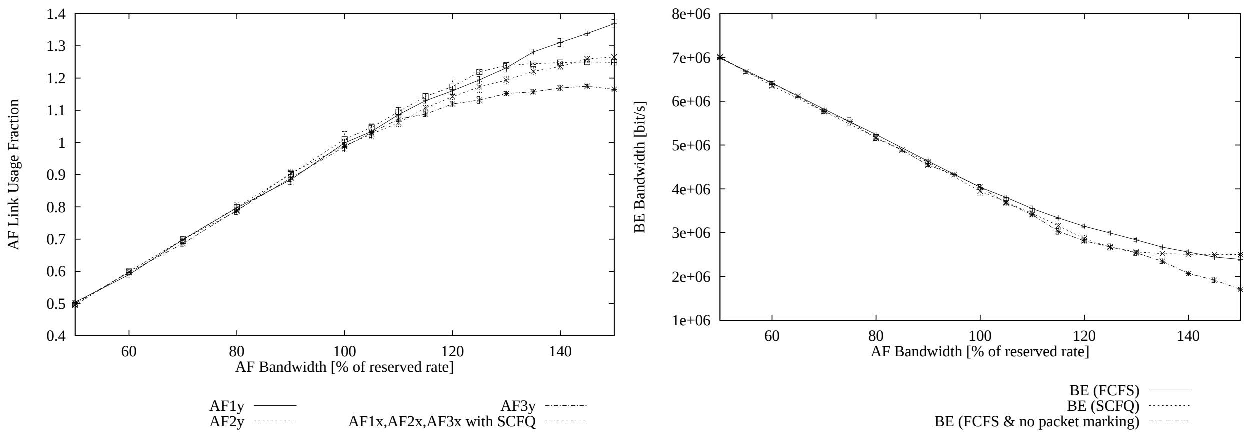

Fig. 7. *Comparing AF and BE bandwidth for FCFS and SCFQ scheduling.*

• Delay-insensitive queues get a high per-queue threshold to profit from unused buffer space.
• For classes that contain adaptive flows, the minimum per-queue threshold is set to half the maximum per-queue threshold. Non-adaptive flows that do not react to early packet drops have equal minimum and maximum per-queue thresholds.
• Drop precedences can be set equal over a set of queues. The higher the per-flow threshold, the lower the drop probability. Setting the per-flow threshold to 1 disables packet marking for this flow, but packets can still be dropped in a severely overloaded network.

V Simulation Results

The simulations described here have been made in a Diffserv-enabled network environment. Flow-and-queue threshold-based buffer management maps well to the Diffserv classes [8], [15] for the following reasons:
• For a system to scale well, the number of queues is important. It is clear that the flow-and-queue threshold-based buffer management does not treat micro-flows individually, but only applies the service defined for the corresponding service class. Thus, the buffer-management system keeps the queue number low, generally not more than several dozens in a Diffserv environment. Packet classification is based on the Diffserv Codepoint, but other classification rules could also be envisaged.
• Setting a low per-queue threshold assures low packet delay, and bandwidth is guaranteed by the queue weight of the WFQ scheduler. This can be used to implement Expedited Forwarding (EF) per-hop-behavior.
• Using a virtual scheduler, service classes in terms of "better than" or Olympic service [8], which consists of three service classes, namely gold, silver and bronze, can be implemented. The Diffserv Assured Forwarding (AF) per-hop-behavior can be used to identify the service class of a packet.
• Support of drop precedences: Various packet markers have been proposed in the Diffserv working group [16], [17]. These markers use the result of a traffic meter to set the appropriate Diffserv Codepoint (DSCP). They should not be confused with packet marking as introduced in this paper. The marking strategy proposed here differs from these Diffserv markers because it acts only locally in a router. Per-flow thresholds are assigned to Diffserv drop precedences in AF to fulfill dropping differentiation. Although packets are only either marked or not, this is sufficient to support multiple levels of drop precedences. The DSCP is not modified in the process, but can influence the marking done by the buffer management.

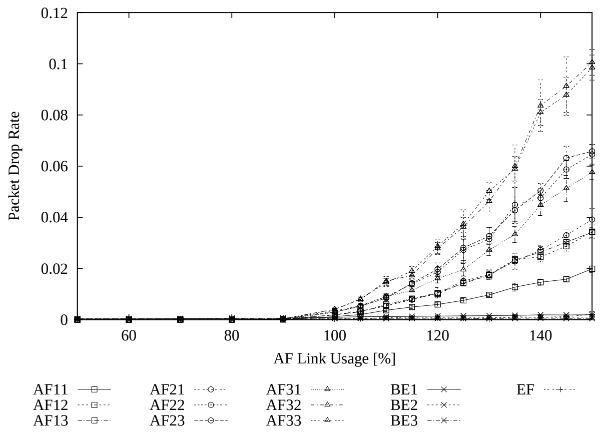

Fig. 8. *Packet drop rate for various traffic classes using a FCFS scheduler.*

V-A *Service Differentiation for Assured Forwarding without Virtual Scheduler*

In this Section FCFS and SCFQ schedulers without a virtual scheduler are compared. These two scheduler types have been chosen to discuss their main properties when combined with DRD and to show that these properties cannot be maintained at the same time. The topology of the simulation is shown in Figure 5, where multiple sources as

TABLE I
Traffic sources.

Flow	*Sources*	*Rate* [% of reserved rate]
EF	Telnet Sources	
AF1y	CBR & Pareto On/Off	from 50% to 150%
AF2y	CBR & Pareto On/Off	from 50% to 150%
AF3y	CBR & Pareto On/Off	from 50% to 150%
AF3y	CBR & Pareto On/Off	from 50% to 150%
BE	greedy TCP sources	

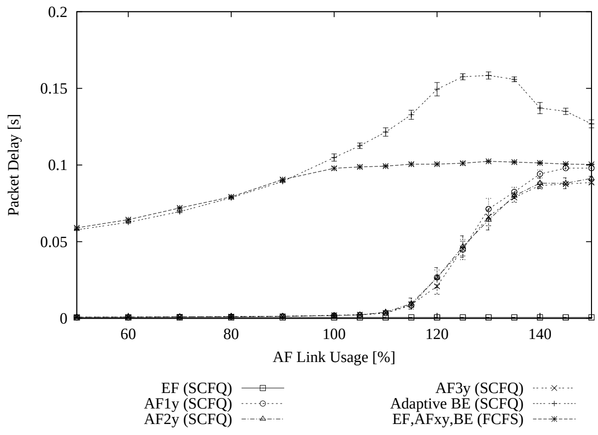

Fig. 9. *Comparing packet delays.*

given in Table I share the same outgoing link at a router. The router uses the DRD scheme as explained in Section III-D. The first queue is assigned to an EF Diffserv class. Ten Telnet applications generate the traffic for this flow. This traffic is substantially lower than the reserved rate, and other flows may borrow from this unused bandwidth. The following three queues treat three AF Diffserv classes: AF1, AF2 and AF3. These flows are generated by CBR and multiple Pareto on/off sources, which create an equal number of all three drop precedences in each AF class. The sources have been chosen such as to be extremely bursty. The average sending rates for all three AF Diffserv sources are equal and vary from 50 to 150% of the allocated bandwidth. The highest-numbered queue is designated for adaptive best-effort (BE) traffic. A set of greedy TCP connections generates this traffic. All queues have equal weights and, therefore, equal reserved bandwidth. All links are set to 10 Mbit/s. The maximum buffer space is set to 160 kBytes. During the simulation, all sources are sending data at the rates given in Table I.

The buffer settings are shown in Table II and Figure 6. The per-queue thresholds are set to guarantee a maximum delay for each class. The terrassing of the per-flow thresholds within an AF class is important to realize drop precedences. The thresholds AFx1, AFx2 and AFx3 are the

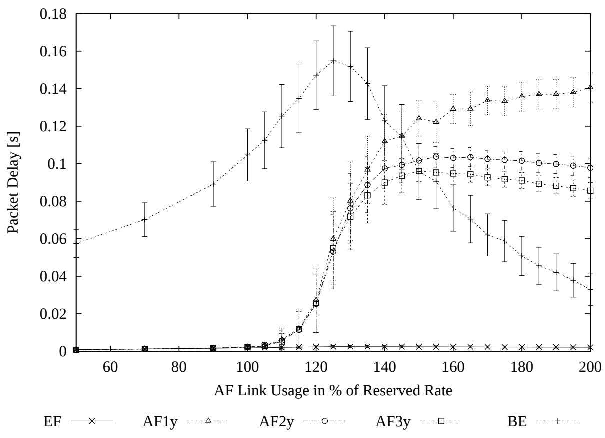

Fig. 10. *Packet delays for different classes.*

same for all AF queues. When setting up the thresholds, no differentiation among the same drop precedence of different AF classes has to be performed. All AF classes start dropping packets at the same per-queue limit. The best-effort RED threshold is set to 40% of its per-queue threshold to enable early congestion avoidance even when the buffer space has been almost completely filled up by other sources.

TABLE II
Buffer thresholds.

Flow	*Thresholds*	
	Per-Flow	*Per-Queue*
EF	1.0	0.2
AFx1	1.0	
AFx2	0.8	0.4
AFx3	0.6	
BE	0.4	1.0

The main goal of introducing packet marking as mentioned in Section III-C is to support service differentiation in the form of AF drop precedences and to improve overall buffer usage. Packet marking does not influence packet order, and packets belonging to the same traffic class will leave the router in the same sequence as they arrived.

The results shown in Figure 7 illustrate the differentiation among AF classes when FCFS or SCFQ scheduling is used. With SCFQ the scheduler completely dominates the bandwidth allocation. Minimum-bandwidth guarantees for best-effort traffic can be given with both schedulers if packet marking is used. Without packet marking, best-effort traffic starts oscillating and loses reserved bandwidth even with SCFQ scheduling.

With the given per-flow thresholds, every AF class is split into three drop precedences (Figure 8). Because of the high network load (when AF classes are sending more than 120% of the reserved rate), the buffer space of a router is

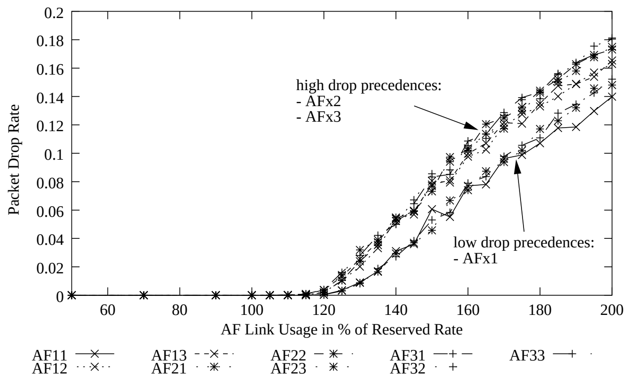

Fig. 11. *Packet drop rate for all AF classes including the drop precedence.*

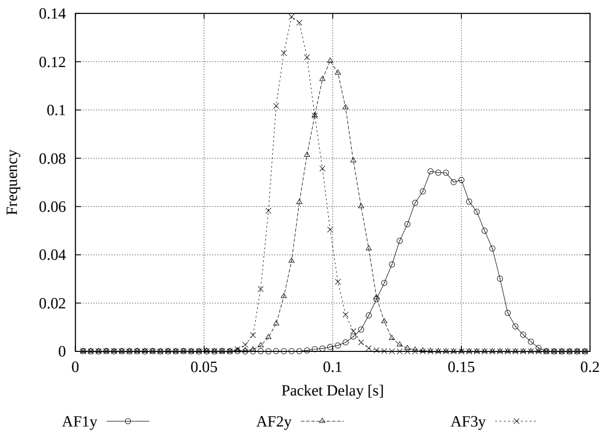

Fig. 12. *Delay distribution for AF classes at 200% of reserved rate.*

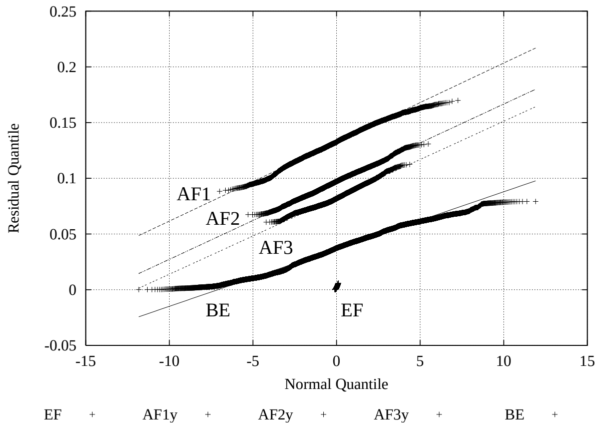

Fig. 13. *Quantile-to-quantile plot for all delay distributions (AF classes sending at 200% of reserved rate). The straight line indicates a linear least-squares fit.*

almost completely filled up at any time of the simulation, and the third per-flow threshold is too low to take effect. Nevertheless the Diffserv requirement of having at least two drop levels is satisfied. Relative service differentiation in terms of packet drop rates is clearly visible.

The packet delays are shown in Figure 9. For scheduling, SCFQ scheduler takes the packet arrival time as well as the amount of packets being stored in a queue into account, whereas in a FCFS scheduler all queues experience the same average delay because FCFS cannot distinguish among the queues. Therefore with FCFS scheduling, the average delays for traffic classes other than best-effort are shifted towards the best-effort values when the actual AF bandwidth is lower than the reserved rate. However, delays have an upper bound given by the per-queue thresholds. This is not the case for FCFS scheduling, and only overall buffer occupancy influences packet delay. To be more precise, decreasing per-queue thresholds would lower overall buffer usage because packets have already been dropped earlier to avoid congestion, and would have an equal effect on packet delays in all queues. On the other hand, we have seen that a WFQ scheduler imposes its fairness properties in a way that traffic differentiation is only feasible through static threshold settings. Although the average delay can be kept within an acceptable range, no significant delay differentiation can be realized with FCFS. Non-best-effort delays are always larger with FCFS for sources using less than their full share of bandwidth [18]. Here packet delay could be improved by using a virtual scheduling algorithm or a “weak” WFQ scheduler that allows higher-priority packets to bypass others.

V-B AF for Gold, Silver and Bronze Services using a Virtual Scheduler

Above, buffer settings to support Diffserv have been introduced and tested using only one scheduler. To facilitate comparison of the results shown above with those described below, the same settings have been used but a virtual scheduler has been added. Again, the incoming traffic for the AF classes stems from CBR and multiple Pareto on/off sources. The AF sending rate range has been increased to 200% of the reserved rate in order to show that even with severe oversubscription tail drops are rare.

The above-described results show how packet drop rate differentiation can be achieved with FCFS while packet delay remains the same for all packets traversing the router. The virtual scheduler scheme has been introduced to overcome this weakness.

Figure 10 shows the average packet delays for all queues. Surprisingly, what was better in terms of drop precedence in simple DRD now becomes worse in terms of delay: This means that the packet delay is shorter in higher-numbered AF classes and therefore AF3x has the best performance in terms of delay. As AF itself does not specify any particular relationship between AF per-hop-behaviors, the AF numbering introduced earlier will be kept. In addition it can be seen that the delay is bounded for each class separately. A intuitive explanation of this result is that DRD combined with a virtual sched-

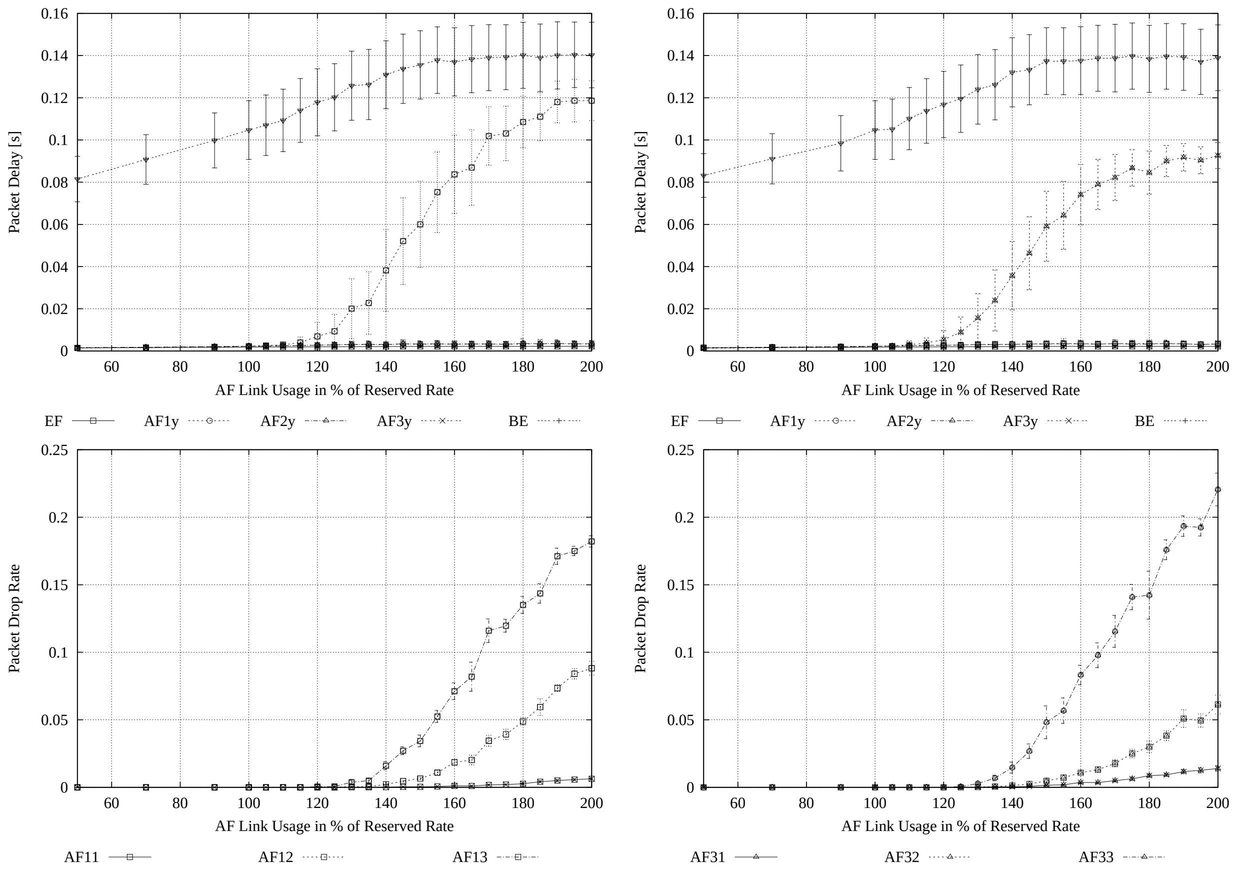

Fig. 14. *Packet delay and drop rate while only one AF class varies.*

uler will start dropping packets earlier in higher-numbered AF classes because of the higher DRD drop probability of those classes, whereas the real scheduler maintains the fair bandwidth allocation and therefore assures equal drop rates for equal incoming traffic.

In contrast to the simple DRD case, with FCFS packet drop rates are similar for all AF classes. Again only two levels of drop precedence in one class are visible (Figure 11). It will be shown later that this is only the case when all classes send at the same rate.

The results clearly show that the virtual scheduler scheme is able to differentiate packet delays while giving a strict bandwidth guarantee according to the configured weight. Excess bandwidth is distributed according to the queue weights.

V-C Delay distribution

Packet delays are distributed approximately normally, as shown in Figures 12 and 13. The latter is a quantile-to-quantile plot, in which the straight line indicates a linear least-squares fit. The slightly S-shaped plots indicate that the distribution is peakier and has shorter tails than a normal distribution. This stems from the fact that delays cannot be negative and that overall buffer space is limited. The delay differentiation is due to the intrinsic behavior of the scheme rather than to sudden queue flushes or other undesired effects. In addition to the lower delay, the AF3y class also has a smaller delay variance, making it an attractive candidate for a “better than” service class.

V-D Varying incoming traffic for one AF class

In the preceding simulation, the AF sending rate has been varied for all AF classes. Now the rate is fixed to 100% of the reserved rate, and only one class at a time varies from 50% to 200%.

Figures 14 show the packet delay and the drop rate when only one incoming rate (AF1y or AF3y) varies. As expected, it confirms the results obtained in Section V-A. In addition, a clear drop precedence differentiation between all three precedences, which has disappeared in the previous simulation, is now distinguishable again. This lets us presume that delay might depend on the number of queues in the system. This has not been tested in this paper, and is left for future work.

V-E Comparison to the basic RED algorithm

In a network environment with severe oversubscription and thus offered loads that exceed the transmission capac-

TABLE III

RED vs. DRD with virtual scheduler.

	Tot. offered Rate	*Early drops*	*Tail drops*
RED	11.98 Mbit/s	44639	92459
DRD	11.71 Mbit/s	120074	2391

ity by far, RED has turned out to be insufficient for efficient congestion indication if the number of TCP connections is high or traffic does not behave in a TCP friendly way. Here we compare DRD congestion avoidance using a virtual scheduler with traditional RED. The total offered load for the simulation has been set to 120% of the available rate. Packet drops are counted during the 100 seconds of simulation time. Table III shows that forced packet drops, known as tail-drops, have been significantly reduced using DRD and virtual scheduler, and is less than 2% of all dropped packets. RED drops more than two thirds of all dropped packets because of buffer overflow. The conclusion is that DRD with virtual scheduling has a excellent potential for efficient early congestion avoidance.

V-F Packet tags

The new scheme needs more memory, as already mentioned, mainly because additional packet tags have to be stored. First of all, it has to be shown that the amount of additional tags is bounded. The set of tags in a queue i is given as T_i, and the subsets of real and virtual tags are T_i^r and T_i^v. The use of a second scheduler leads to an overall increase of packet tags in the system. The set of extra tags is written as $T_i^e = T_i^v \setminus (T_i^r \cap T_i^v)$. If only a real scheduler is used then $T_i = T_i^r$, otherwise i.e. with a virtual scheduler, $T_i = T_i^r \cup T_i^v$. Figure 15 shows that under heavy load in the AF1 queue $|T_{AF1} \setminus T_{AF1}^r| \rightarrow 0$ and $T_{AF1}^v \subset T_{AF1}^r$, whereas in the AF3 queue $|T_{AF3} \setminus T_{AF3}^v| \rightarrow 0$ and $T_{AF3}^r \subset T_{AF3}^v$. The consequence is that for the former the set of extra tags is $|T_{AF1}^e| \rightarrow 0$ and for the latter $|T_{AF3}^e| \rightarrow |T_{AF3}^v| \neq 0$, causing the increase in the total of packet tags.

We found that

$$2|T^r| \geq |T| \qquad (4)$$

holds for all offered loads. As compared to a packet these tags are small, the impact on overall memory increase is justifiable.

VI Conclusion

In this paper we introduced a two-threshold-based buffer-management system that can be used for relative service differentiation in Diffserv AF per-hop-behaviors. The main new parts are the DRD congestion avoidance scheme, internal packet marking, and a virtual scheduler. The DRD congestion avoidance scheme enables dynamic and relative service differentiation even with a simple scheduler such as FCFS. The fact that no delay differentiation is possible when used with FCFS led to the introduction of a virtual scheduler scheme. By means of simulations, it has been shown that a virtual scheduler is a robust management scheme for heavy and bursty traffic load. In conjunction with DRD, the scheme is able to perform relative delay differentiation of AF Diffserv per-hop-behavior while guaranteeing minimum bandwidth and fair

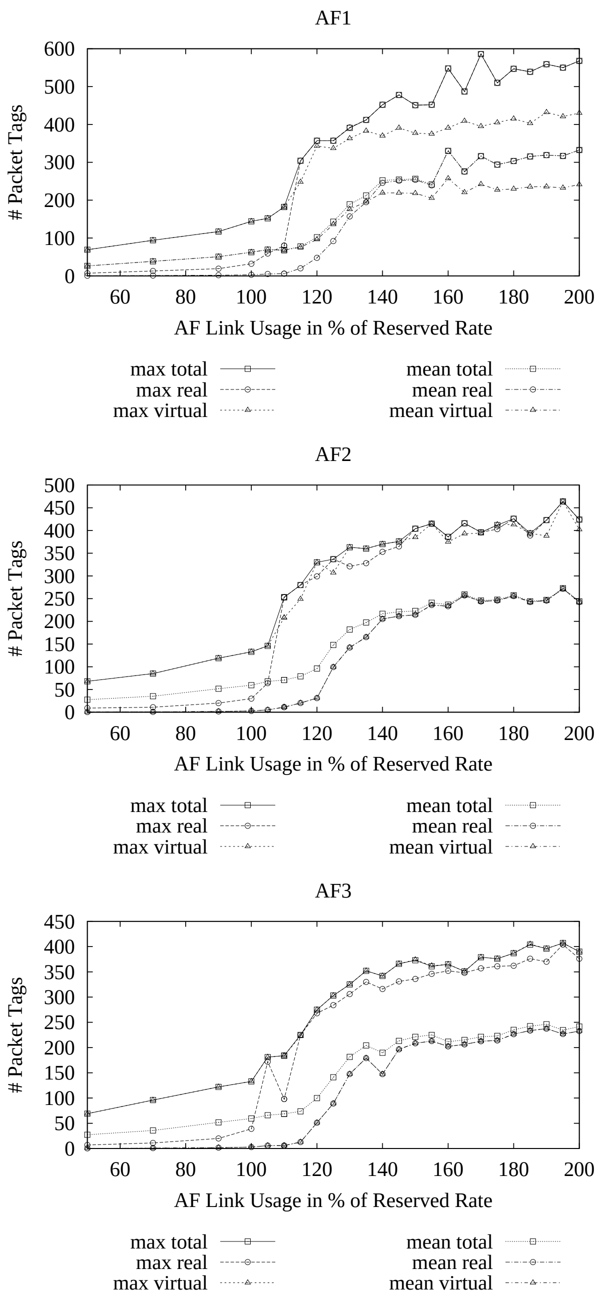

Fig. 15. *Comparing AF and BE bandwidth for FCFS and SCFQ scheduling.*

excess bandwidth allocation. Moreover the scheme avoids tail drops and, therefore, does not lead to TCP synchronization effects. Compared to other schemes, DRD with a virtual scheduler uses only few parameters (per-queue and per-flow thresholds, queue priority and queue weight) that are set at initialization time, and then requires no further tuning.

Packet marking is an important enhancement to flow-and-queue threshold-based buffer-management systems that allows the implementation of at least two drop precedences within a queue. In addition to optimizing overall buffer usage, packet marking is even necessary to avoid bursty packet drops. The influence of responsive and non-responsive flows in the same queue can have a significant impact on inter-flow fairness, but would exceed the scope of this paper and is left for future work.

References

[1] S.J. Golestani. A Self-Clocked Fair Queuing Scheme for Broadband Applications. *ACM Computer Commun. Rev.*, April 1994.

[2] A.K. Parekh and R.G. Gallager. A Generalized Processor Sharing Approach to Flow Control. June 1993.

[3] D. Stiliadis and A. Varma. Design and Analysis of Frame-Based Fair Queueing: A New Traffic Scheduling Algorithm for Packet-Switched Networks. May 1996.

[4] J. C.R. Bennett and H. Zhang. WF^2Q: Worst-case Fair Weighted Fair Queueing. 1996.

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

[6] R. Pletka, P. Droz, and R. Haas. A New Buffer Management Scheme for IP Differentiated Services. Technical Report RZ 3216 (# 93262), IBM Research, Zurich Research Laboratory, March 2000.

[7] S. Blake, D. Blake, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Services, RFC2475, December 1998.

[8] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. Assured Forwarding PHB Group, RFC2597, June 1999.

[9] S. Bajaj, L. Breslau, D. Estrin, K. Fall, S. Floyd, P. Haldar, M. Handley, A. Helmy, J. Heidemann, P. Huang, S. Kumar, S. McCanne, R. Rejaie, P. Sharma, K. Varadhan, Y. Xu, H. Yu, and D. Zappala. Improving Simulation for Network Research. Technical Report 99-702b, University of Southern California, March 1999.

[10] C. Drovolis, D. Stiliadis, and P. Ramanathan. Proportional Differentiated Services: Delay Differentiation and Packet Scheduling. *ACM Computer Commun. Rev. (SIGCOMM '99)*, September 1999.

[11] F. Risso. Decoupling Bandwidth and Delay Properties in Class Based Queueing. *Dipartimento di Automatica e Informatica, Politechnico di Torino*, 2000.

[12] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. *ACM Trans. on Networking*, August 1993.

[13] R. Guérin, S. Kamat, and V. Peris. Scalable QoS Provision Through Buffer Management. October 1998.

[14] D. Clark and W. Fang. Expicit Allocation of Best Effort Packet Delivery Service. *ACM Trans. on Networking, Vol. 6, No. 4*, August 1998.

[15] V. Jacobson, K. Nichols, and K. Poduri. An Expedited Forwarding PHB, RFC2598, June 1999.

[16] J. Heinanen and R. Guerin. A Single Rate Three Color Marker, RFC2697, September 1999.

[17] J. Heinanen and R. Guerin. A Two Rate Three Color Marker, RFC2698, September 1999.

[18] A. Demers, S. Keshav, and S. Shenker. Analysis and Simulation of Fair Queuing Algorithm. *ACM Computer Commun. Rev., pp. 3-12*, 1989.