Research Report

A new Buffer Management Scheme for IP Differentiated Services

R. Pletka, P. Droz, R. Haas

IBM Research
Zurich Research Laboratory
8803 Rüschlikon
Switzerland

IBM Research
Almaden · Austin · Beijing · Delhi · Haifa · T.J. Watson · Tokyo · Zurich

A new Buffer Management Scheme for IP Differentiated Services

R. Pletka, P. Droz, R. Haas

IBM Research, Zurich Research Laboratory, 8803 Rüschlikon, Switzerland

Abstract

The sophisticated Quality-of-Service (QoS) demands of research, education and commercial network service providers require new services in current "best-effort" Internet architecture. The Internet must enable applications that demand specific services to profit from a set of differentiated traffic classes, which support either relative or absolute types of quality of service, or both. This paper focuses on the design of scalable buffer management and queueing strategies in a QoS-enabled Internet. A threshold-based buffer management to be used mainly in core routers is proposed and evaluated. A new buffer management scheme, called Differentiated Random Drop (DRD) scheme, is introduced. Combined with simple first-come, first-served (FCFS) scheduling, the scheme can support differentiated services (Diffserv) that is being standardized by the IETF.

1   Introduction

In a QoS-enabled network we distinguish between privileged and best-effort flows. The easiest way for a router to treat such types of traffic is to use multiple queues, which contain flows, called microflows, or aggregates thereof called macroflows. Flows can be easily mapped to Diffserv [1] traffic classes.

Unfortunately today's bandwidth increase does not allow one to keep state for each microflow in a router. Flexible solutions that scale well are required. To achieve sufficient scalability, relative simple functionality will be placed in core routers, whereas the more complex control operations such as classifying and aggregating will take place in edge routers.

Factors influencing QoS in a router are scheduling and buffer management. Packet scheduling refers to the decision process used to choose which packets should be sent next. Scheduling influences the order of service. Fairness and the computational complexity characterize each scheduling algorithm. In the past ten years, considerable research has been done to improve packet-scheduling algorithms [2, 3, 4, 5].

Buffer management refers to any particular discipline used to regulate the occupancy of a queue or the entire buffer space. The regulation takes place by admitting packets to use buffer space and dropping "unwelcome" packets. An intelligent buffer management scheme can allow one to provide rate guarantees to flows even with schedulers as simple as FCFS [6].

A recently proposed simple, efficient and scalable queue management algorithm is Approximative Longest Queue Drop (ALQD) [7]. Compared with Random Early Detection (RED) [8], its main goals are fairness in bottleneck link bandwidth sharing, overall throughput improvement, and isolation and protection from aggressive sources that consume more than their fair share of bandwidth.

Although ALQD is very scalable, the scheme alone poorly meets Diffserv's needs: Whereas RED starts dropping packets before the total buffer space is full, ALQD waits until no more space is available. This leads to a hard limit, which can cause bursty packet drops. Bursty packet drops are even worse when all dropped packets are from the same flow. Adaptive flows such as TCP will then stall. Supposing there were one or more longest queues in a buffer system, ALQD would only drop packets from these queues without performing any type of congestion avoidance to other flows.

Introducing additional random characteristics may solve the problem of ALQD. There should be a RED-like mechanism, which starts dropping earlier with a given (increasing) probability. The queue from which a packet should be dropped may be chosen randomly by using a dynamic per-queue probability. The dynamic per-queue probability can be evaluated using the average queue size. This gives longer queues a higher probability to be chosen. It is even conceivable that packets should be dropped not only from the longest queue but from a set of longest queues with a dropping probability corresponding to their queue length.

Section 2 introduces a flow-and-queue threshold-based buffer management scheme and shows various extensions thereof. These extensions are fundamental to fair packet dropping and better overall buffer usage. In order to avoid bursty packet drops and traffic oscillations caused mainly by TCP synchronization, The use of a robust and efficient congestion-avoidance scheme is essential.

In Section 3 two packet dropping schemes are explained in which the experience gained with ALQD leads to a new congestion-avoidance scheme: the Differentiated Random Drop (DRD) scheme. This scheme is able to provide differentiated services with a simple first-come, first-served (FCFS) scheduler.
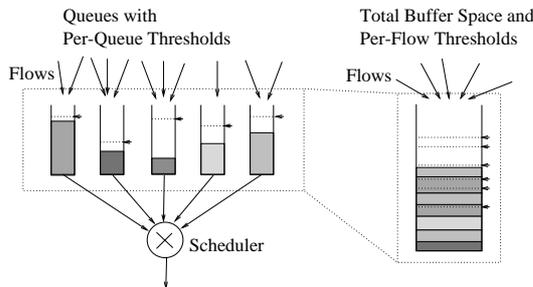
Figure 1: *Architecture of buffer management.*

Section 4 explains why such a congestion-avoidance scheme has to be used in a flow-and-queue threshold-based buffer management scheme. The simulations are done with the network simulator *ns* [9] to which specific new extensions have been added.

The two congestion-avoidance schemes introduced in Section 3 are compared in Section 5. A FCFS as well as a weighted fair queueing (WFQ) scheduler are used for the comparison. The results show that the DRD scheme with FCFS enables dynamic and adaptive drop rates, which are relative to the other queues in the system. The drop rate between queues can be expressed in terms of "better than" as proposed in the Diffserv framework.

The simulations discussed in Section 6 show how different drop precedences in the same queue can be implemented when a simple packet marking as described in Section 2.3 is used.

2 Threshold-Based Buffer Management

To support QoS in a buffer management system we introduce as shown in Figure 1 a threshold-based mechanism called flow-and-queue threshold-based buffer management scheme. Thresholds are assigned to macroflows and queues. Macroflows may consist of a large number of microflows that form a flow aggregate. For simplicity we call flow aggregates just flows from now on. Each of these flows is attributed to one queue. This process maps well to the Diffserv classes [10, 11].

The number of queues in a system is important in order to scale well. The flow-and-queue threshold-based buffer management system keeps this number low, in general not more than several dozen in a Diffserv environment.

The first threshold limits a flow's global buffer occupancy 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 of packets. Marking and dropping depends on the type of buffer management and will be discussed later.

The second threshold is a **per-queue threshold**, which allows a segmentation of the available buffer space. When the per-queue threshold is exceeded, packets have to be dropped in order to bind the maximal packet delay. The per-queue threshold acts as a "hard" dropping policy when used with no additional strategies. Hard means that packets may be suddenly dropped in bursts. This behavior is not at all desirable. An early dropping policy such as RED should be combined with this threshold.

In this scheme it is possible to allocate more than the real existing buffer space to queues as opposed to hard segmented buffer spaces as in [6]. This means that the sum of all per-queue thresholds may exceed the total available buffer space. The advantage of such a strategy is
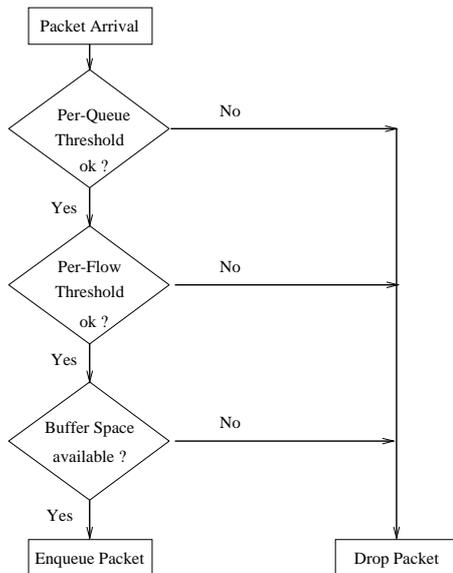
Figure 2: *Hard dropping scheme.*

that it supports larger bursts of a flow when other flows are on a low buffer usage level or even not backlogged and therefore uses less global buffer space. On the other hand when all flows are sending at peak rate at the same time the fixed per-queue buffers cannot be fully exploited. At this point the per-flow threshold will act as a limiter. As we will see below, this is even needed to give best-effort traffic the capability to take advantage of unused bandwidth.

It is clear that the buffer management does not treat microflows individually, but only applies the service defined for the corresponding service class. In a similar way, this corresponds to a Diffserv-enabled core router that does not distinguish among microflows for the routing process.

2.1 Hard dropping scheme

The simplest known buffer management scheme consists of dropping packets when no more buffer space is available. This strategy turns out to be inadequate for performing efficient and fair packet forwarding even when used with fair queueing. Packets are often dropped in "bursts" from a single flow, whereas other flows increase their traffic even more.

Adding a flow-and-queue threshold-based buffer management allows buffer sharing and priority handling to be provided. In addition to dropping when no bufferspace is available, packets are dropped when one or both of the two thresholds are exceeded (Figure 2). The simulations discussed in Section 4 show that a simple flow-and-queue threshold-based buffer management as described above 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 avalanches once threshold is exceeded. There must be some additional packet-dropping strategies to avoid burst drops and to avoid synchronization of TCP sources.
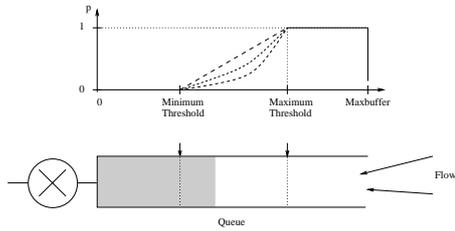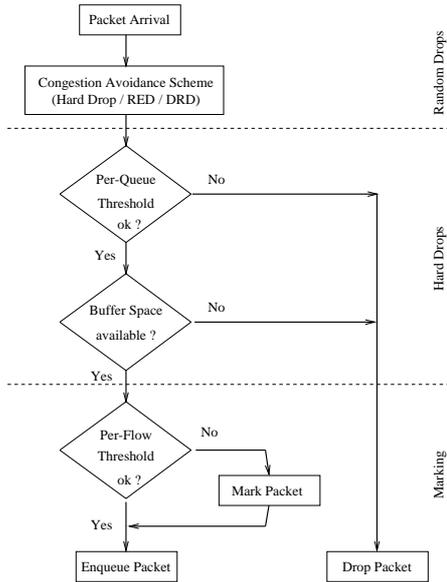
Figure 3: *Examples of probability functions.*

Figure 4: *Dropping scheme with congestion avoidance and packet marking.*

2.2 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 3). We used a linearly increasing function because of its simplicity. To use a linearly increasing function, it is sufficient to add a threshold. The two thresholds together are then used as a lower and upper limit.

The size of the linearly increasing section has to be set relatively to the global buffer space or to the allocated queue space. Too small a value does not overcome the bursty drop problem and too large a section will introduce too early and unnecessary packet drops. Without going into more details concerning the optimal setting, which is beyond the scope of this paper, experience has shown that a value of around 50% is reasonable [8].

2.3 Introducing packet marking

Various packet markers have been proposed in the Diffserv working group [12, 13]. These markers use the result of a traffic meter to set the appropriate Diffserv Codepoint (DSCP). The marking strategy proposed below is different from these markers and acts only inernally. In fact, the DSCP is not modified in the process, but it influences the marking done by the buffer management scheme as described in Section 6. As mentioned earlier there may be multiple flows with different per-flow thresholds within the same queue. Marked packets in a

queue may belong to different flows without making any flow distinction between them when drop decisions have to be made. The per-flow drop rate increases with decreasing per-flow threshold.

It is easy to see that a small per-flow threshold excludes a flow completely from getting 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 the 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 on, the marked ones should be dropped first. Figure 4 shows where packet marking is done in our scheme. As a result buffer space is used more efficiently and more packets are served. 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. The congestion-avoidance scheme that preferably drops marked packets is discussed in the next section.

3   Congestion Avoidance

In a hard-dropping scheme, packets are often dropped in bursts from a single flow, whereas other flows increase their traffic even more. To avoid this, a widely used mechanism to indicate congestion has been introduced by Floyd [8] called RED. Routers detect incipient congestion by computing the average queue size of each connection. The dropping probability is a linearly increasing function of the average queue size starting at a given threshold. Routers can notify connections of congestion either by dropping or marking packets. A special mechanism to avoid global TCP window synchronization is included.

Unfortunately this dropping scheme does not scale to a very large number of connections because the state of each connection must be kept and maintained by routers.

In addition, connections that do not adapt their bandwidth when congestion is indicated, such as UDP traffic, will benefit from the adaptive character of TCP as shown in [14]. These non-adaptive sources should be shaped at the network edge, treated in separate queues, or marked as best-effort traffic.

As line speed increases a dropping scheme should be as simple as possible and preferably implemented in hardware in order to avoid CPU usage in routers.

3.1   RED drop scheme

A special congestion-avoidance scheme can prevent undesirable burst drops. This in hardware-coded algorithm is executed upon packet arrival. Instead of using RED on each connection it can be applied to a queue containing flows as illustrated in Figure 1. When a packet arrives, RED is evaluated for the queue where the packet will be enqued and, if necessary, a packet is discarded from this queue's head. The packet-dropping probability is calculated as a linear function of the queue size. Then the per-queue and per-flow thresholds as well as the available global buffer space are examined before the packet can be enqueued (Figure 4). Owing the resemblance to RED we call this the RED scheme. The main characteristic of the RED scheme is to combine congestion avoidance and fairness amongst queues and flows.

In a network where QoS is supported, different dropping probabilities should appear in

different traffic classes. Flows with stronger needs of QoS should suffer less from packet drops than best-effort traffic. The RED scheme does not differentiate among queues because it depends only on the average queue size. Differentiation among queues has to be done in a static manner when setting up the queue parameters. Therefore a more sophisticated congestion-avoidance scheme has to be introduced: We call it simply the Differentiated Random Drop (DRD) scheme.

3.2   Differentiated random drop (DRD) scheme

The main goal of differentiated random drop scheme is to introduce a dynamic per-queue drop probability while adding relative dependency among the various queues in the system. This will allow service differentiation such as "better than" or Olympic service, which consists of three service classes: gold, silver and bronze.

   The scheme works as follows: When a packet arrives, the following congestion-avoidance mechanism is performed before processing of that packet continues as explained in Section 3.1. One of the queues is chosen randomly using a dynamic per-queue probability. Then random early discard is 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. The per-queue probability is proportional to the amount of bytes in the queue plus the amount of bytes in all higher-priority queues. This is a more general approach as is used for RIO in [15]. Queues containing no packets have zero per-queue probability. It should be noted that priorities are introduced only for dropping behavior and not, for example in CBQ [16], as a per-queue priority used for scheduling purposes. The per-queue probability p_i can be written

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

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

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

and the constant C is then given by

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

4   Fair Packet Dropping

To illustrate the difference between the various dropping schemes mentioned before we consider the following simulation where multiple incoming links have a maximum rate of 10 Mbit/s each and variable link delays share the same outgoing link. The outgoing link has a rate of only 1 Mbit/s (Figure 5). For the simulation, multiple traffic sources with different characteristics and different on/off times were taken and accumulated to flows as listed in Table 1. Although the reserved rates of these flows do not correspond to actual Internet backbone traffic, similar traffic mixture can be encountered in a QoS-enabled network. Here we use these sources only to show the impact of different dropping schemes on other flows present in the router. Flow 0 has a strong need for QoS and should not be influenced by other flows. It can be modeled
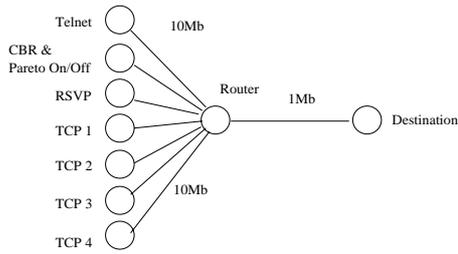
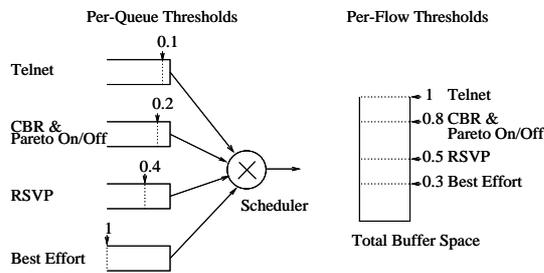Figure 5: *Structure of the simulation.*
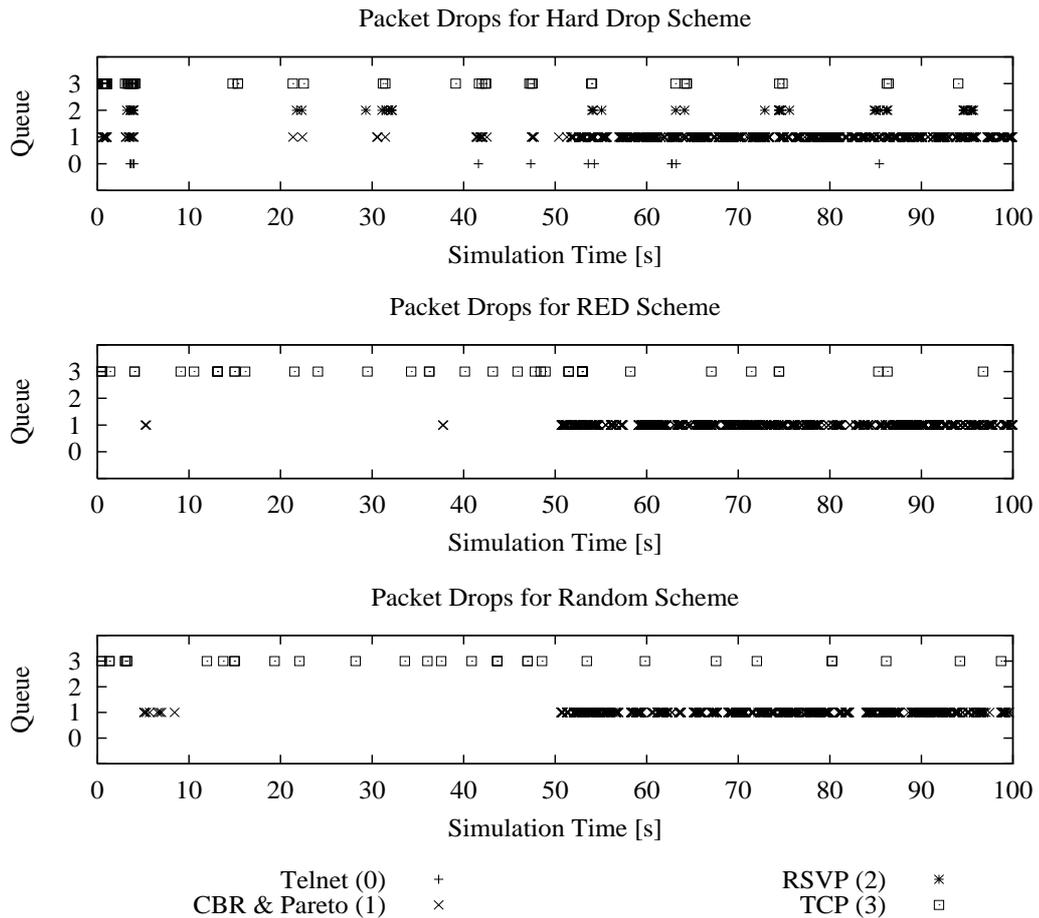

Figure 6: *Threshold settings.*

Figure 7: *Packet drops in Hard Drop, RED and DRD scheme using SCFQ scheduler.*

Table 1: *Sources.*

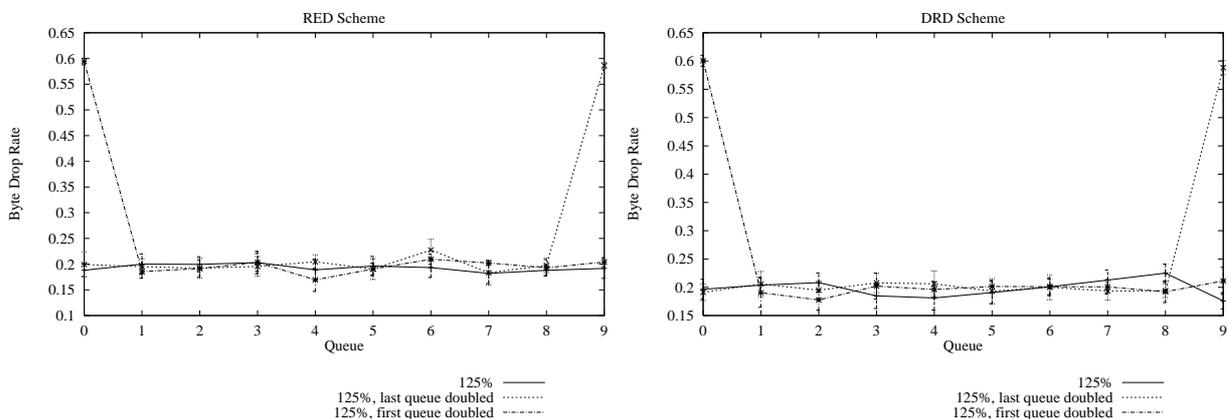| Flow	Sources	Rate [kbit/s]		Time [s]
		sending	*reserved*	
0	Telnet		100	2.5-100
1	CBR	200	250	0-100
	Pareto on/off	200		0-15
	Pareto on/off	500		35-100
2	RSVP Flow	500	500	3-10,
				16-35,
				55-100
3	4 greedy TCP sources	150		0-100

Figure 8: RED and DRD scheme with SCFQ scheduler.

as an Expedited Forwarding (EF) Diffserv class [11]. Flow 1 starts slightly overbooked from $t \in [0, 15]$s sending at 300 kbit/s when only 250 kbit/s are reserved. This does not become a problem because flow 0 is not using its full rate. Flow 2 is an RSVP flow that sends during three intervals, and the last flow consists of several greedy TCP sources. The overall buffer space available is 50 kByte. All thresholds are shown in Figure 6 as a fraction of the total buffer space.

Figure 7 shows packet drops for the "hard drop", RED and DRD schemes from this simulation using SCFQ scheduling [3]. Hard drop punishes even flows that do not exceed their reserved rate (Telnet traffic and RSVP flow in the present case). In addition, bursty drops are frequent. The RED and DRD schemes only drop packets from highly filled queues, e.g. from flows that significantly exceed their reserved rate and therefore need some kind of congestion avoidance. Bursty drops can no longer be found. The difference between the RED and DRD schemes is discussed in the next section.

5 Comparing the RED and DRD schemes

In this section the RED and the DRD schemes are compared. The structure of the simulation is similar to the previous one given in Figure 5 except for the traffic sources. Ten traffic classes with equal thresholds (the per-queue thresholds are set to 10% and the RED thresholds to 5% of the global buffer space) and equal bandwidth reservation are introduced. Each of these
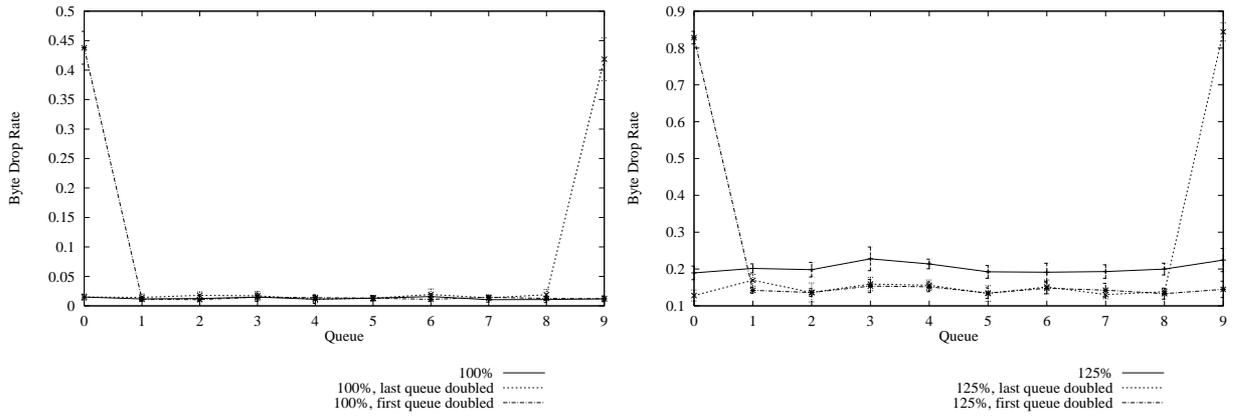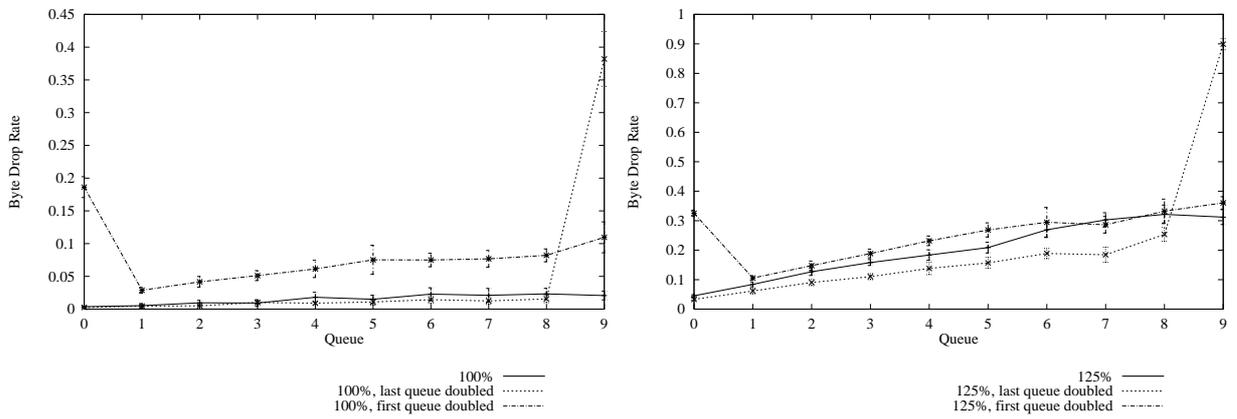
Figure 9: RED scheme with FCFS scheduler.

Figure 10: DRD scheme with FCFS scheduler.

classes maps into one queue. Inside a class, traffic is generated using a set of ten Pareto on/off sources. In the DRD scheme the difference between queues is given by the increasing queue election probability. For the RED scheme there is no difference between the queues. The outgoing link at the router is set to 10 Mbit/s. Incoming sources send at 100% and 125% of the reserved bandwidth. In a second and third simulation the first or the last queue's sending rate is doubled to observe the influence on other queues. The results in Figures 8, 9 and 10 are given with a confidence interval of 90%. SCFQ and FCFS are used as scheduling algorithms. The per-flow threshold is set to the maximum buffer space. Its effect is not examined in this simulation.

When SCFQ is used, the congestion-avoidance scheme does not influence the behavior under heavy load, and the scheduling algorithm completely dominates the drop rate (Figure 8). Because of the high incoming charge, queues always contain packets to transmit and the SCFQ scheduler assigns the same bandwidth share (given by the queue weights) for all queues. All spare packets will be dropped sooner or later, independently of the dropping scheme used. The question is now whether the congestion-avoidance scheme provides differentiation of QoS for simple FCFS scheduling.

As expected the RED dropping scheme with FCFS does not differ when the last or first queue is overfilled (Figure 9). When incoming traffic is about 125% of the total outgoing link speed, even more packets than expected are dropped in the overfilled queues with doubled incoming rate. In general queues with doubled incoming rate are more severely punished than the other ones. Using the RED dropping scheme and FCFS no relative priorities between queues are possible. Differentiation between queues can only be achieved when per-queue thresholds are modified. But this modification affects the maximum delay guarantee for the queues which is not desired. In addition such a differentiation is static because it has to be set with the thresholds at initialization time.

The DRD scheme has a completely different behavior with FCFS (Figure 10): Higher priority queues experience fewer packets dropped in the overloaded situation. In other words, when a customer would like to have an Assured Forwarding (AF) service [10], it can be argued that only half the packets are dropped in the highest priority queue [1] than in the lowest one for the same overload. It is clear that queues with lower priority suffer from this overload. The additional drop rate is distributed on all lower priority queues. The other way round, a queue with lower priority does not influence negatively the drop rate of queues with higher priority. The DRD scheme allows relative QoS with FCFS. To implement an EF Diffserv class, the highest priority queue must be used in order to get the appropriate service.

Until now the per flow threshold has not been exploited. In the next section we will show how it can be used to achieve AF drop precedences in a Diffserv network.

6   Implementing drop precedences using packet marking

The main goal of introducing packet marking as mentioned in Section 2.3 is to support service differentiation in the form of AF [2] drop precedences and to improve overall buffer usage. Packet

---

[1] The term "highest priority queue" is somewhat misleading here: It does not mean that a fixed priority is assigned to a queue but refers to the algorithm used in the congestion-avoidance scheme for choosing a queue where a packet will be dropped. The term "election priority" that indicates the order of preference would be more appropriate.

[2] We use the following notation for the AF per-hop-behaviours: Flow of AF class x and drop precedence y is written as AFxy.
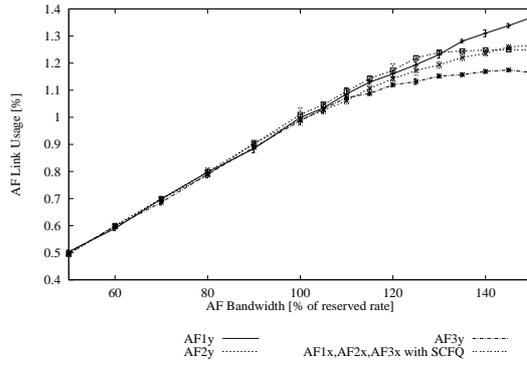
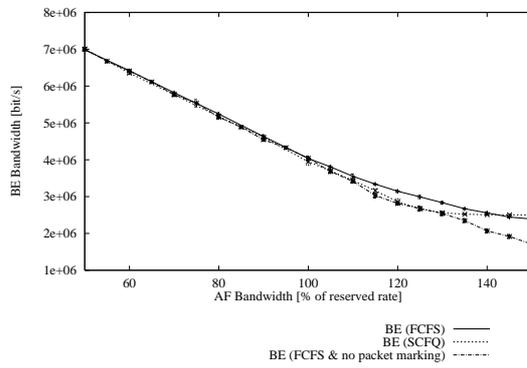Figure 11: *Comparing AF bandwidth for FCFS and SCFQ scheduling.*

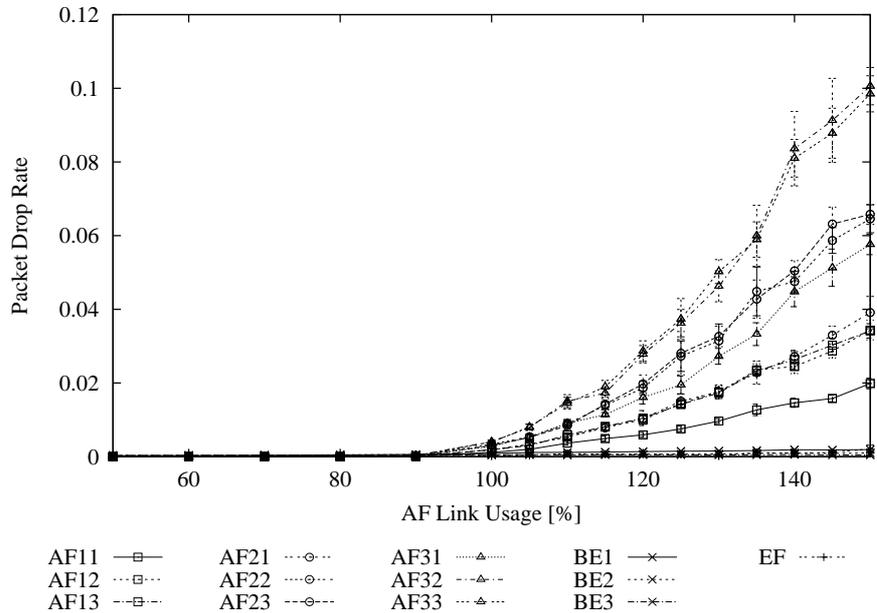Figure 12: *Comparison of best-effort (BE) bandwidth.*

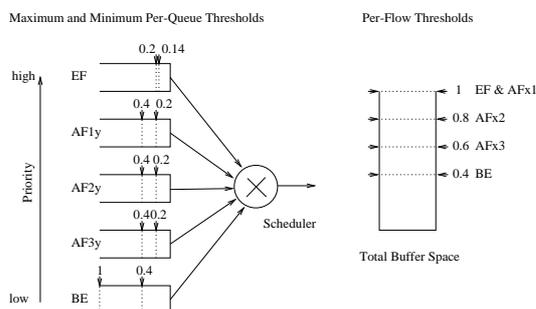Figure 13: *Packet drop rate for different traffic classes using a FCFS scheduler.*

Table 2: *Sources.*

Flow	*Sources*	*Rate*
		[% of reserved rate]
EF	10 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	6 greedy TCP sources	

Figure 14: *Per-Flow and Per-Queue thresholds.*

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 topology of the simulation is almost the same as in Section 5 where multiple sources share the same outgoing link at a router and only the traffic sources have been changed (Table 2). The router uses the DRD scheme as explained in Section 3.2 with FCFS and SCFQ scheduling; its outgoing link is set to 10 Mbit/s. The first queue is assigned to an EF Diffserv class. Ten Telnet applications generate the traffic for this flow. This traffic is a lot less than the reserved rate. The following three queues treat three AF Diffserv classes: AF1y, AF2y and AF3y. The flows are generated by CBR and multiple Pareto on/off sources. The average sending rates for all three AF Diffserv sources are equal and vary from 50 to 150% of the allocated bandwidth. The last queue is designated for adaptive best-effort traffic. A set of greedy TCP connections generates this traffic. All queues have equal weights and therefore equal reserved bandwidth. The maximum buffer space is set to 160 kBytes. During simulation time all sources are sending.

The buffer settings are shown in Table 3 and Figure 14. The per-queue thresholds are set to guarantee a maximum delay for each class. Important is the terrassing of the per-flow thresholds within an AF class to realize drop precedences. The thresholds AFx1, AFx2 and AFx3 are the same for all AF queues. No differentiation among the same drop precedence of different AF classes has to be done when setting up the thresholds. 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 is almost completely filled up by other sources.

The results shown in Figure 11 illustrate the differentiation amongst AF classes when FCFS scheduling is used. With SCFQ the scheduler dominates completely the bandwidth allocation. Minimum bandwidth guarantees for best-effort traffic can be given with both
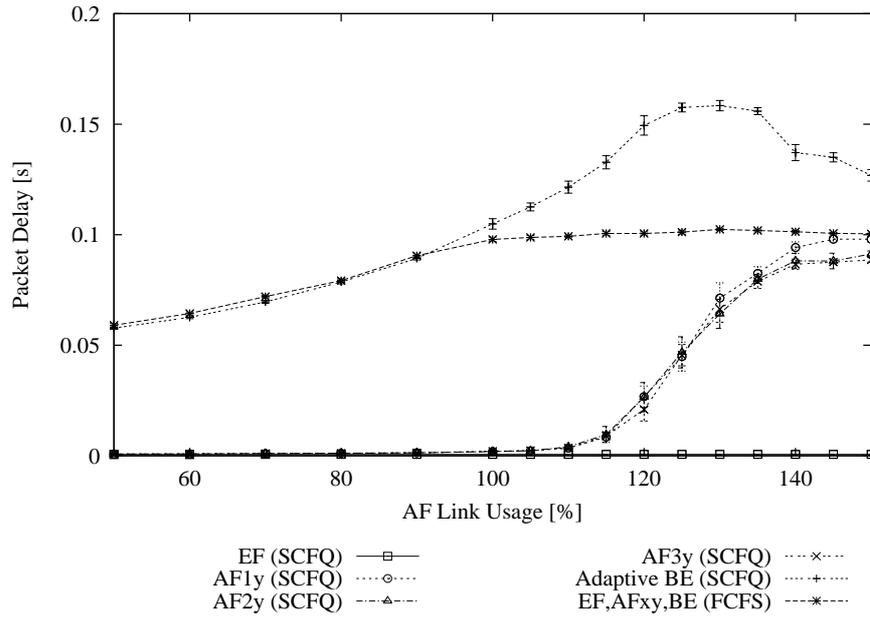
Figure 15: *Comparing packet delays.*

Table 3: *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

schedulers if packet marking is used (Figure 12). Without packet marking, best-effort traffic starts oscillating and losing reserved bandwidth even with SCFQ scheduling.

With the given per-flow thresholds every AF class is split into three drop precedences (Figure 13). Because of the high network load (when AF classes are sending more than 120% of the reserved rate) the router's buffer space is 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 requirements are satisfied.

The packet delays are shown in Figure 15. A SCFQ scheduler takes the packet arrival time as well as the amount of packets being stored in a queue into account for the scheduling, whereas in a FCFS scheduling all queues experience the same average delay because FCFS cannot distinguish amongst the queues. Therefore with FCFS scheduling, the average delays for traffic classes other than best-effort are pulled 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. 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 [17]. Here packet delay could be improved while using a "weak" WFQ scheduler, that allows higher-priority packets to bypass others.

7 Conclusion and Outlook

In this paper we have described a new congestion-avoidance scheme for offering dynamic and relative service differenciation in a router. The scheme is tested in a flow-and-queue threshold-based buffer management system. We have shown that with a simple FCFS scheduler and a sophisticated packet-dropping scheme, traffic differentiation is feasible and efficient enough to support Diffserv, especially the AF classes. Minimum bandwidth guarantee and fair excess bandwidth allocation are supported. In addition the scheme scales well, which makes it interesting for use in core routers.

The new packet marking scheme is an important improvement on the flow-and-queue threshold-based buffer management system which allows the implementation of various drop precedences within a queue. Not only overall buffer usage can be optimized but it is even necessary to avoid bursty drops. However the admission operations take place in the network edge, which treats responsive and nonresponsive flows in the same queue can have a significant impact on interflow fairness. This aspect could be the topic of future work.

References

[1] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, *An Architecture for Differentiated Services, RFC 2475*, December 1998

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

[3] S.J. Golestani, *A Self-Clocked Fair Queueing Scheme for Broadband Applications*, April 1994

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

[5] J. C.R. Bennett, H. Zhang, *WF²Q: Worst-case Fair Weighted Fair Queueing*, 1996

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

[7] B. Suter, T.V. Lakshman, D. Stiliadis, A. Choudhury, *Efficient Active Queue Management for Internet Routers*, November 21, 1997

[8] S. Floyd, V. Jacobson, *Random Early Detection Gateways for Congestion Avoidance*, August 1993

[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, D. Zappala, *Improving Simulation for Network Research. Technical Report 99-702b, University of Southern California*, March, 1999. revised September 1999, to appear in IEEE Computer

[10] J. Heinanen, F. Baker, W. Weiss, J. Wroclawski, *Assured Forwarding PHB Group, Internet Draft*, February 1999

[11] V. Jacobson, K. Nichols, K. Poduri, *An Expedited Forwarding PHB, Internet Draft*, February 1999

[12] J. Heinanen, R. Guerin, *A Three Color Marker, Internet Draft*, February 1999

[13] H. Kim, *A Fair Marker, Internet Draft*, April 1999

[14] S. Floyd, K. Fall, *Router Mechanisms to Support End-to-End Congestion Control*, February 1997

[15] D. Clark, J. Wroclawski, *An Approach to Service Allocation in the Internet, Internet Draft*, July 1997

[16] S. Floyd, V. Jacobson, *Link-sharing and Resource Management Models for Packet Networks*, August 1995

[17] A. Demers, S. Keshav, S. Shenker, *Analysis and Simulation of Fair Queuing Algorithm*, 1989