Random Early Marking: Improving TCP Performance in DiffServ Assured Forwarding

Sandra Tartarelli and Albert Banchs
Network Laboratories Heidelberg, NEC Europe Ltd.

***Abstract*— In the context of Active Queue Management, intelligent dropping algorithms have been proposed to achieve a better link utilization with TCP. In this paper we apply this research to the problem of achieving a better utilization of a customer's contracted throughput when it is sending a TCP traffic aggregate. We propose a scheme, Random Early Marking (REM)[1], that improves the throughput of a TCP aggregate by early marking some packets as *out*. The proper configuration of REM has been analyzed from a control theoretical standpoint. Simulation results show that REM leads to a significant improvement in a wide range of environments.**

***Index Terms*— Differentiated Services, Assured Forwarding, Random Early Marking, Token Bucket, guaranteed throughput, TCP**

I. Introduction

The Differentiated Service (DiffServ) architecture [1] has been proposed as a scalable way of providing Quality of Service (QoS) in the Internet. Scalability is achieved by moving complicated functionality toward the edge and leaving the core with very simple functionality. With DiffServ, packets are marked at the ingress of the network with a DiffServ codepoint (DSCP) and at the core they are given a forwarding treatment according to their DSCP. Each DSCP corresponds to a Per-Hop Behavior (PHB).

The IETF's DiffServ Working Group has defined two PHB groups: the Expedited Forwarding (EF) PHB [2] and the Assured Forwarding (AF) PHB [3]. This paper focuses on the latter. Typically, an ISP would use the AF PHB to provide a service where the packets of a customer are forwarded with a very high probability as long as the aggregate traffic from the customer does not exceed the target rate he has contracted, the Committed Information Rate (CIR).

Even though the AF PHB has become a standard, there are still some open issues that need to be understood. The most important question relates to the kind of end-to-end services that can be provided [4], [5]. There is the concern that AF should show some measure of predictability for paying customers. If charging is based on the CIR, then in a predictable system a customer would expect to receive a throughput at least equal to the CIR.

Unfortunately, in case the customer is sending an aggregate of TCP traffic, current schemes proposed for AF do not always show this predictable behavior. TCP traffic increases its sending rate steadily and decreases it upon detecting packet drops. In case of congestion, AF drops customer's packets when his total sending rate exceeds his CIR. In some situations, the combination of AF and TCP results in a synchronized behavior of the customer's TCP sources, that decrease all their sending rate at the same time. As a consequence, the customer's sending rate is oscillating, which results in a lower throughput than the CIR.

In the context of queue management in routers, numerous Active Queue Management (AQM) schemes (see e.g. [6], [7]) have been proposed to fight TCP synchronization in order to achieve a better link utilization. These schemes are based on the idea of performing some intelligent early dropping before the buffer is completely filled up.

In this paper we take up the research done in the context of AQM and apply it to the problem of achieving a better utilization of the customer's CIR. Specifically, we propose to intelligently mark the customer's packets such that TCP sources are randomly early notified of congestion and synchronization is avoided. Our performance objective is to guarantee a throughput to a customer sending a TCP traffic aggregate the closest possible to his CIR.

The rest of the paper is structured as follows. In Section II we propose our scheme, *Random Early Marking*, to improve the performance of TCP in AF. Section III analyses the proposed scheme from a control theoretical standpoint, and uses this analysis to give guidelines on its configuration. We present our simulation results in Section IV and conclude the paper with a summary in Section V.

II. Random Early Marking

In this paper we assume that the Assured Forwarding PHB is implemented with the WRED algorithm [8] with two levels of drop precedence (*in* and *out*). The proposed scheme, however, could be easily extended to three levels of drop precedence (*green*, *yellow* and *red*).

WRED works as follows. Packets entering the network are marked *in profile* or *out of profile* at the ingress according to the Token Bucket algorithm of Figure II. In this algorithm, a token bucket of depth B is filled at the rate specified by the CIR, where B and CIR are part of the contract with the customer. Then, when a packet from the customer is received, it is marked *in* if there are enough bytes in the bucket for this packet, and it is marked *out* otherwise. In case of *in* marking, a number of bytes equal to the packet length is substracted from the bucket.

At core nodes, all packets, *in* and *out*, are put into the same buffer. This buffer is managed in such a way that in case of congestion *out* packets are dropped first. In this way, the WRED algorithm guarantees that, as long as the network is configured such that *in* packets alone do not cause congestion, *in* packets are never dropped. In [9] we provide configuration guidelines to achieve this behavior.

[1] The Random Early Marking scheme has a patent application pending on it.

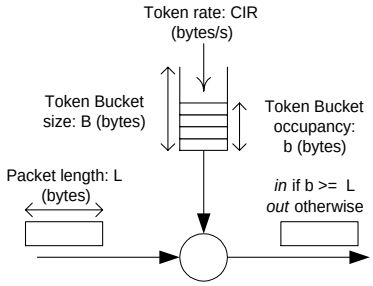

Fig. 1. Token Bucket algorithm.

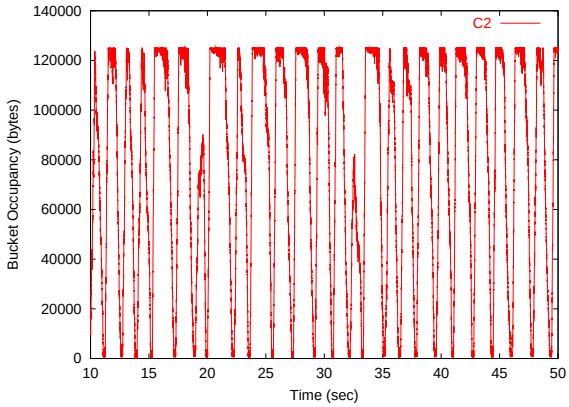

Fig. 2. Token Bucket occupancy for customer 2.

It can be easily seen that, in the case of a customer sending non-responsive UDP traffic, WRED achieves the goal of guaranteeing to a customer a throughput at least equal to the CIR he has contracted. This is because, as long as the customer is not sending above the CIR, the token bucket is never emptied and all packets are marked *in*.

Providing throughput guarantees for TCP traffic with WRED, though, is more complex. While no packet drops are detected, TCP traffic increases steadily its sending rate. When the sending rate reaches the CIR, the token bucket is emptied and packets are marked *out*. This *out* marking leads to drops in case of congestion. TCP then reacts to these drops by decreasing its sending rate to a lower value than the CIR. As a consequence, TCP's sending rate oscillates between the CIR and a lower value, which results in an average throughput lower than the committed.

The behavior of TCP described above is better illustrated with the following example. Two customers who have contracted a 10 Mbps CIR share a 20 Mbps link. Both customers are sending 20 TCP flows each and have RTTs of 20 (customer 1) and 100 ms (customer 2). According to simulation results, the throughputs obtained by customers 1 and 2 in this scenario are 9.83 and 8.32 Mbps respectively. We can observe that the throughput obtained by customer 2 is considerably lower than his CIR.

Figure 2 plots the occupancy of the token bucket for customer 2. The plot shows the oscillating behavior of his TCP traffic aggregate. When the token bucket gets empty it is because the TCP traffic has increased its rate over the CIR. In case of congestion, some low priority packets are dropped by the WRED mechanism. TCP reacts to the drops by significantly decreasing its rate. At this point, the token bucket fills up again. It is not until TCP increases its rate over the CIR again that the token bucket occupancy decreases. In the time period while the bucket is full the customer is transmitting at a lower rate than the CIR.

The objective of providing a throughput equal to the CIR can be reformulated as stabilizing the token bucket occupancy to a value smaller than the bucket size. A constant not full token bucket occupancy implies a sending rate of *in* packets equal to the CIR. Since *in* packets are very unlikely to be dropped, this leads to the committed throughput.

Let b be the token bucket occupancy, B its size, $r(t)$ the customer's sending rate and C the contracted CIR. Then the problem of stabilizing the token bucket occupancy b can be expressed as having the time derivative of the token bucket occupancy ($\dot{b}$) equal to 0:

$$\dot{b} = C - r(t) = 0 \tag{1}$$

with $0 < b < B$.

The above problem is similar to the problem of stabilizing the occupancy q of a queue of size B and capacity C filled at a rate $r(t)$. Actually, assuming constant round-trip delays and that all *out* packets are dropped, the two problems are equivalent, which can be easily seen with the change of variable:

$$q = B - b \tag{2}$$

The problem of stabilizing the buffer occupancy has been extensively studied in the context of Active Queue Management (AQM), where numerous schemes have been proposed in the last few years. While these schemes differ in details, they are similar at the architectural level. They monitor the evolution of the buffer occupancy and process this data with an algorithm to obtain a dropping probability for incoming packets. Different AQM schemes basically differ in the algorithm used to obtain the dropping probabilities.

In this paper we use the algorithms proposed in the context of AQM to stabilize the occupancy of the token bucket. The AQM algorithm that we have chosen to use in this paper is the *Proportional Integrator* (PI) proposed in [7].

Hence, the *Random Early Marking* (REM) scheme we propose is based on using the PI algorithm of [7] with the change of variable defined above (Equation 2) to determine the probability p of marking an incoming packet as *out*. This leads to the pseudocode of Algorithm 1 for computing p. Note that when marking *out*, no bytes are substracted from the token bucket. Note also that when the token bucket is empty, packets are marked *out* independently of p.

Algorithm 1 REM pseudocode

For every incoming packet:
$p = k1(b\_ref - b) - k2(b\_ref - b\_old) + p\_old$
$p\_old = p$
$b\_old = b$

The stability of the token bucket occupancy with Algorithm 1 depends on the parameters $k1$ and $k2$. b_ref is the desired

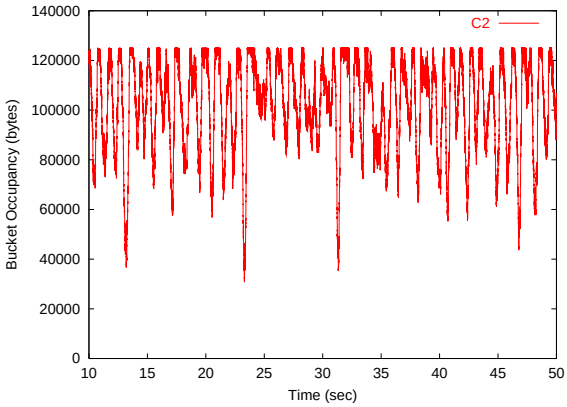

Fig. 3. Token Bucket occupancy with REM.

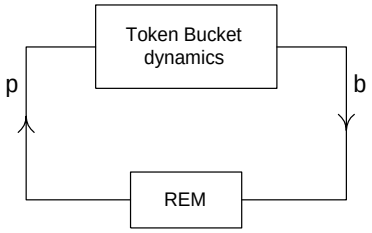

Fig. 4. REM feedback control system.

buffer occupancy to which we want to regulate. Therefore, the appropriate choice for $k1$ and $k2$ is key to achieve the performance objective of REM. In the following section we study how to configure these parameters.

Figure 3 corresponds to the example of Figure 2 but using REM with the configuration of Section III. It can be observed that REM stabilizes the token bucket occupancy around $b\_ref = 0.75B$. The throughput obtained by customer 2 in this case is 9.65 Mbps, which is much closer to the CIR than the 8.32 Mbps obtained without REM. Note that in Figure 3, as compared to Figure 2, the time intervals over which the token bucket is full are shorter.

III. Analysis and Configuration

In this section we apply the control theoretic analyses of [7] and [10] to the problem of stabilizing the token bucket occupancy in REM.

Figure 4 shows a feedback control system depiction of REM. The action of REM is to mark packet *out* (with probability p) as a function of the token bucket occupancy b.

In the rest of this section we assume the worst possible scenario for the throughput, in which all *out* packets are dropped. With this assumption, the dependency of the token bucket occupancy b on the probability of marking *out* p is expressed with the following two equations [11]:

$$\begin{aligned} \dot{W} &= \frac{1}{R(t)} - \frac{W(t)W(t-R(t))}{2R(t-R(t))}p(t-R(t)) \\ \dot{b} &= -\frac{W(t)}{R(t)}N(t) + C \end{aligned} \tag{3}$$

where W is the TCP window size, $R(t)$ is the round-trip time, $N(t)$ is the number of TCP sources and C is the contracted CIR.

Taking (W, b) as the state and p as the input, the operation point (W_0, b_0, p_0) is defined by $\dot{W} = 0$ and $\dot{b} = 0$ so that

$$\begin{aligned} W_0^2 p_0 &= 2 \\ W_0 &= \frac{R_0 C}{N} \end{aligned} \tag{4}$$

Assuming $N(t) = N$ and $R(t) = R_0$ as constants, we linearize (3) to obtain

$$\begin{aligned} \delta\dot{W} &= -\frac{N}{R_0^2 C}(\delta W + \delta W(t-R_0)) - \frac{R_0 C^2}{2N^2}\delta p(t-R_0) \\ \delta\dot{b} &= -\frac{N}{R_0}\delta W \end{aligned} \tag{5}$$

where

$$\begin{aligned} \delta W &= W - W_0 \\ \delta b &= b - b_0 \\ \delta p &= p - p_0 \end{aligned}$$

Performing a Laplace transform on the above differential equations leads to the block diagram of Figure 5 for the linearized REM control system.

With the assumption that $\frac{N}{R_0^2 C} \ll \frac{1}{R_0}$ (this assumption is justified in [10]), $H(s)$ can be written as

$$H(s) = -\frac{R_0 C^2}{2N^2}\frac{1}{s + \frac{2N}{R_0^2 C}}e^{-sR_0} \tag{6}$$

$C(s)$ is the Laplace transform on the controller of Algorithm 1:

$$C(s) = K\frac{\frac{s}{z}+1}{s} \tag{7}$$

where K and z are functions of the parameters $k1$ and $k2$.

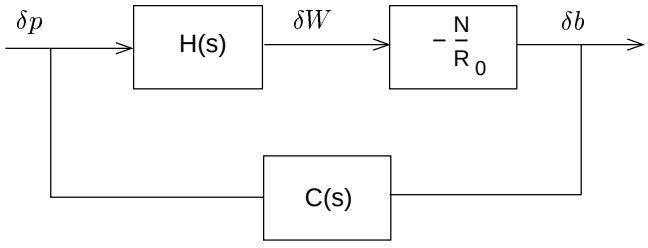

Fig. 5. Block diagram of linearized REM control system.

The above gives us the following open loop transfer function:

$$L(jw) = e^{-jwR_0} \cdot \frac{C^2 K}{2N} \cdot \frac{\frac{jw}{z}+1}{jw} \cdot \frac{1}{jw + \frac{2N}{R_0^2 C}} \tag{8}$$

Let's assume a range for the number of TCP sessions, say $N \geq N^-$, and the round-trip time, $R_0 \leq R^+$. The objective is to select REM values K and z to stabilize the linear control system of Figure 5.

We first choose the following zero for the controller:

$$z = w_g = 0.1\frac{2N^-}{R^{+2}C} \tag{9}$$

The rationale behind the above choice is to have $C(s)$ to dominate the closed-loop behavior. This is done making the closed loop constant ($\approx 1/w_g$) greater than the TCP time constant $\frac{2N^-}{R^{+2}C}$.

Then we invoke the Nyquist stability criterion [12], which states that if $\mid L(jw_g) \mid \leq 1$ and $\angle L(jw_g) > -180^o$ then the system is stable at w_g. By imposing $\mid L(jw_g) \mid = 0.1$ we obtain the following value for K:

$$K = 0.007\frac{(2N^-)^3}{(R^{+2}C^2)^2} \tag{10}$$

and computing $\angle L(jw_g)$ with the above conditions we obtain

$$\angle L(jw_g) \geq -146^o > -180^o \tag{11}$$

Finally, transforming $C(s)$ from the s domain (Laplace transform) back to the z domain by using the bilinear transformation yields the following configuration rules for $k1$ and $k2$:

$$k1 = K\left(\frac{T}{2} + \frac{1}{w_g}\right)$$
$$k2 = -K\left(\frac{T}{2} - \frac{1}{w_g}\right) \tag{12}$$

where K and w_g are defined in Equations 9 and 10, and T is the packet interarrival time, which we have taken as $T = 1/C$ (i.e. we assume that the customer is transmitting at his CIR).

IV. Simulation Results

In [9] we proposed some rules to configure a DiffServ scenario, that uses a token bucket and a WRED queue. These rules proved very efficient in providing the agreed CIR in many simulated scenarios.

However, we observed that in a number of cases of interest in the practice, such an architecture is not able to contrast fairness problems due to the TCP flow control mechanism. In this section we show that by employing REM, results can be significantly improved. We configured the simulation scenarios following the guidelines given in [9]. In particular, the discarding thresholds for conforming traffic are set to a value that avoids *in* packets drops. Besides we set $OUT_{max} = 10$ (maximum threshold for *out* packets). Finally, for simulations with the REM mechanism, we considered the instantaneous queue length for the AQM mechanism, so that the system reacts faster to the early marking. Simulations were run by using ns-2 [13].

In the following we first show some simulation results we obtained by considering a number of heterogeneous scenarios. In the first three cases we assumed a fully subscribed link, i.e. the sum of the CIRs is equal to the bottleneck capacity, while in the fourth example we explored the behavior when the link is only partially subscribed. We then conclude the section, by evaluating the performance of the proposed marking scheme as a function of different parameters. For all simulations we considered TCP Reno.

A. Scenario 1: TCP-UDP interaction

The first scenario we considered is described by Table I and also depicted in Figure 6. The access links do not introduce either delay or packets drops.

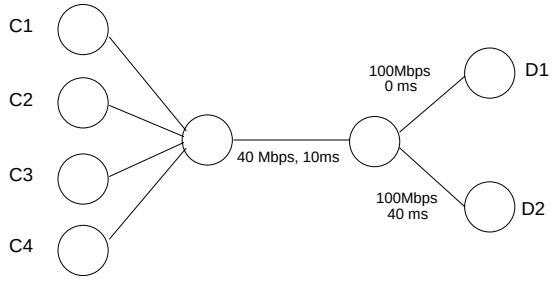

Fig. 6. Simulation scenario.

It is known that non-responsive UDP traffic causes problems of fairness when interacting with TCP flows. Therefore, in this scenario we study the interaction between customers transmitting either TCP only flows or mixed TCP and UDP traffic. To model UDP traffic, we considered Constant Bit Rate (CBR) flows, each sending at 1.5 Mbps. In this case the UDP rate sums up to 75% of the agreed CIR.

	CIR (Mbps)	# of Flows TCP+UDP	RTT (ms)	noREM (Mbps)	REM (Mbps)
Total	-	-	-	37.88	39.47
C1	10	10 + 0	20	9.46	10.10
C2	10	10 + 0	100	7.99	9.05
C3	10	10 + 5	20	10.35	10.21
C4	10	10 + 5	100	10.06	10.09

TABLE I
SCENARIO 1

Table I reports some settings we selected for this test and in the last two columns it shows the results in terms of throughput, for the standard approach and the proposed scheme respectively. Table I confirms that REM helps customers sending only TCP flows to receive a higher share of the total bandwidth. In particular C1, characterized by a small RTT, achieves the agreed CIR while C2 gets more than 90% of it, against the 80% allowed by the standard approach.

B. Scenario 2: heterogeneous CIR values

A fairness problem arises also when different customers contract heterogeneous values for the CIR. In fact, those customers characterized by a lower CIR value are favored in achieving the agreed CIR. With this scenario we give an example of a possible behavior. The bottleneck link speed is set equal to 22 Mbps. Table II shows that in the considered case, REM allows to improve the overall link utilization by more than 15% and above all it leads to a significantly more fair bandwidth distribution.

C. Scenario 3: large number of customers

When the number of customers and of flows grows to high values, then the multiplexing gain has a positive effect towards

	CIR (Mbps)	# of Flows TCP	RTT (ms)	noREM (Mbps)	REM (Mbps)
Total	-	-	-	18.18	21.62
C1	10	10	20	8.63	10.16
C2	10	10	100	7.07	9.23
C3	1	10	20	1.43	1.16
C4	1	10	100	1.03	1.06

TABLE II
SCENARIO 2

better link utilization and bandwidth distribution, even when the standard token bucket is used. The bottleneck link speed is set equal to 100 Mbps. In Table III we show simulation results that confirm this. However, also in this case, the REM approach seems to slightly improve the overall performance.

	CIR (Mbps)	# of Flows TCP	RTT (ms)	noREM (Mbps)	REM (Mbps)
Total	-	-	-	97.17	98.63
C1	10	40	20	10.36	10.58
C2	10	10	100	9.16	9.25
C3	10	10	20	9.91	10.10
C4	10	40	100	10.11	10.27
C5	10	20	20	10.20	10.33
C6	10	20	100	9.79	9.89
C7	10	15	20	10.11	10.25
C8	10	15	100	9.47	9.66
C9	10	5	20	8.88	9.05
C10	10	10	100	9.14	9.22

TABLE III
SCENARIO 3

D. Scenario 4: undersubscribed link

In the last example, we investigate the interaction among customers with only TCP flows and only UDP flows respectively in an under-subscribed link. We considered a link speed of 53 Mbps, while $\sum_{i=1}^{4} CIR_i = 40$ Mbps (75% subscribed link). C3 and C4 transmit both 10 CBR flows, each at a rate of 1.5 Mbps, i.e. their sending rate is significantly above the CIR.

	CIR (Mbps)	# of Flows TCP+UDP	RTT (ms)	noREM (Mbps)	REM (Mbps)
Total	-	-	-	49.56	51.31
C1	10	10+0	20	11.34	14.30
C2	10	10+0	100	9.72	10.48
C3	10	0+10	20	14.25	13.44
C4	10	0+10	100	14.24	13.08

TABLE IV
SCENARIO 4

Table IV shows that REM allows TCP to obtain a significantly higher share of the excess bandwidth as compared to the standard approach. This is especially true for more aggressive TCP customers (C1 has a smaller RTT), while C2 having a relatively small number of flows and a large RTT (respectively 10 and 100ms) can only achieve the agreed CIR.

E. Number of customers

Scenario 3 showed that if the number of flows per customer is high enough, the number of customers is large and if the network is properly configured, then the target throughput is approximately achieved by all customers, even when not using early marking.

In the following we intend to investigate the benefit offered by the REM approach as a function of the number of customers. To this end, we considered again the setting implemented for Scenario 3 and we evaluated the throughput achieved respectively by C1 and C2 as a function of the total number of customers. C1 is characterized by a low RTT and a large number of flows, therefore it is very likely that it will achieve the agreed CIR. C2 on the contrary has a large RTT and a relatively small number of flows, thus it is penalized in the bandwidth sharing. In the simulations, we always considered the first n customers in Table III for a scenario with n customers.

In Figure 7 we compare the throughput obtained by C1 and C2 when using REM and the standard token bucket. REM always allows to achieve the best performance. However, the most significant improvement is achieved by customer C2 when the total number of customers is below 8. By employing REM, C2 always obtains at least 90% of the committed CIR, while the standard token bucket considerably penalizes it when the total number of customers is low. Note that the latter case might be common for access links.

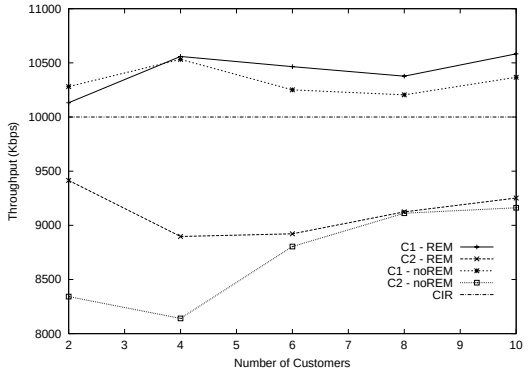

Fig. 7. Achieved throughput as a function of the number of customers.

F. Utilization

For the same experiment, we also evaluated the total link utilization. The results are reported in Figure 8. The improvement due to the REM mechanism is noticeable.

G. Number of flows per customer

We conclude this section, by considering the effect of a low number of flows per customer, of the order of a few units (for instance home users). In particular we analyze the performance of a scenario with 10 customers, each transmitting 10 flows, except for one customer that sends a smaller number of flows. All customers are assigned a CIR=10 Mbps, the RTT varies for the different customers between 20 and 100 ms and the bottleneck speed link is equal to 100 Mbps.

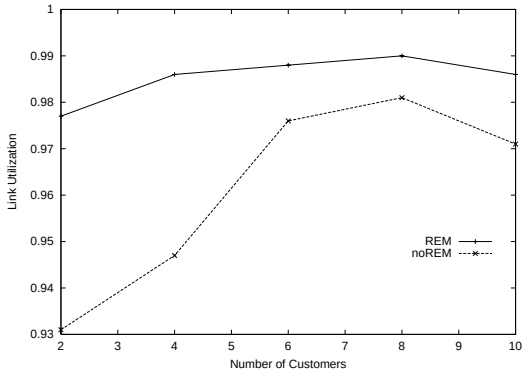

Fig. 8. Achieved link utilization as a function of the number of customers.

We focus on the single customer sending a small number of flows. Based on previous results, we expect it to receive a smaller share of the total bandwidth. We therefore evaluate the achieved throughput as a function of the number of flows, when employing REM as compared to the standard token bucket. Results are reported in Figure 9. As expected, when the number of flows is small, the throughput obtained is significantly lower than the agreed CIR. However, by using REM we observe a relevant improvement. In our example, by transmitting 5 flows, the customer already obtains the agreed CIR, while when no early marking is applied, the throughput achieved is still 10% lower than the agreed one.

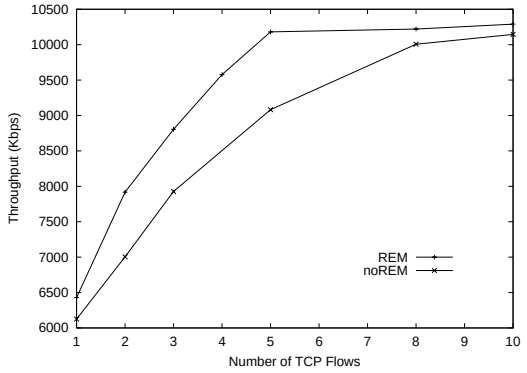

Fig. 9. Achieved throughput as a function of the number of TCP flows.

V. Conclusions

In this paper we have proposed REM (*Random Early Marking*), a mechanism to improve the performance of TCP in DiffServ Assured Forwarding. REM relies on early notifying TCP sources of upcoming congestion via *out* marking. In this way, REM avoids synchronization among the TCP sources of a customer, resulting in a better utilization of the contracted throughput (i.e. a more predictable service) and a better distribution of the total bandwidth (i.e. a higher level of fairness between customers).

REM is integrated into the widely used token bucket algorithm. The only modification introduced by REM to this algorithm is that in some cases packets are marked *out* before the token bucket gets empty. Hence, the good features of the token bucket algorithm for controlling the rate of *in* packets are preserved with REM.

One of the key aspects of REM is its simplicity. Instead of keeping state for each active connection, REM only requires a small number of additional fixed and variable parameters for each token bucket. Another key aspect is that its configuration does not require specific knowledge about the customer's traffic, but only a lower bound for the number of TCP session and an upper bound for the round-trip time.

Simulation results show that the improvement obtained by using REM is notable when one of the following two conditions holds: *a)* the total number of customers traversing a congested link is small or *b)* the number of flows per customer is small. With a large number of customers sending a large number of flows each, results without REM are already good and there is little room for improvement.

We conclude that REM leads to a significant improvement in a wide range of environments, since access links (condition *a)*) are often bottleneck links in the Internet, and home users (condition *b)*) represent a large number of potential customers of the DiffServ AF service.

Acknowledgements

We would like to thank S. Sato, K. Kobayashi and H. Pan for many fruitful discussions on TCP performance in DiffServ. We would also like to thank F. Raspall and M. Molina for useful comments and suggestions on earlier drafts of this paper.

References

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

[2] V. Jacobson, K. Nichols, and K. Poduri, "An Expedited Forwarding PHB," RFC 2598, June 1999.

[3] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured Forwarding PHB Group," RFC 2597, June 1999.

[4] N. Seddigh, B. Nandy, and J. Heinanen, "An Assured Rate Per-Domain Behavior for Differentiated Services," Internet draft, February 2001.

[5] M. Brunner, A. Banchs, S. Tartarelli, and H. Pan, "A one-to-any Assured Rate Per-Domain Behavior for Differentiated Services," Internet draft, April 2001.

[6] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 1, pp. 397–413, August 1993.

[7] C. V. Hollot, V. Mishra, D. Towsley, and W. Gong, "On Designing Improved Controllers for AQM Routers Supporting TCP Flows," in *Proceedings of IEEE INFOCOM 2001*, Anchorage, Alaska, April 2001.

[8] CISCO SYSTEMS, "Congestion Avoidance Overview," `http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fqos_c/fqcprt3/qcfconav.htm`.

[9] S. Sato, K. Kobayashi, H. Pan, S. Tartarelli, and A. Banchs, "Configuration Rule and Performance Evaluation of Diffserv Parameters," in *Proceedings of Seventeenth International Teletraffic Congress (ITC17)*, Salvador da Bahia, Brazil, September 2001.

[10] C. V. Hollot, V. Mishra, D. Towsley, and W. Gong, "A Control Theoretic Analysis of RED," in *Proceedings of IEEE INFOCOM 2001*, Anchorage, Alaska, April 2001.

[11] V. Mishra, W. Gong, and D. Towsley, "Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED," in *Proceedings of ACM/SIGCOMM 2000*, Stockholm, Sweden, August 2000.

[12] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*, Addison-Wesley, 1995.

[13] UCB/LBNL/VINT, "Network Simulator (ns), version 2," `http://www.isi.edu/nsnam/ns/`.