Performance Evaluation of a new Adaptive Packet Marking Scheme for TCP over DiffServ Networks

Giovanni Neglia, Giuseppe Bianchi, Marilena Sottile

Dipartimento di Ingegneria Elettrica, Università degli Studi di Palermo
Email: giovanni.neglia@tti.unipa.it, bianchi@elet.polimi.it, marilena.sottile@libero.it

***Abstract*— In Differentiated Services networks, packets may receive a different treatment according to their Differentiated Services Code Point (DSCP) label. As a consequence, packet marking schemes can be devised to differentiate packets belonging to a same TCP flow, with the goal of improving the experienced performance. This paper presents an extensive performance evaluation of a new adaptive packet marking scheme, applied to a traffic scenario composed of TCP flows with different length. The proposed marking scheme is most efficient when applied to a scenario composed of all long-lived flows. In a realistic mixed traffic scenario, composed of both long-lived and short-lived TCP flows, our marking scheme provides excellent performance in high utilization conditions, and still provides improved performance in medium utilization conditions, comparable with that achieved by a recently proposed marking algorithm specifically devised for short-lived flows. We also propose a "three colors" marking scheme, which merges this approach with ours to achieve improved performance in all utilization conditions.**

I. Introduction

Differentiated Services (DiffServ) networks provide the ability to enforce a different forwarding behavior to packets, based on their Differentiated Services Code Point (DSCP) value. A possible way to exploit the DiffServ architecture is to provide differentiated support for flows belonging to different traffic classes, distinguished on the basis of the DSCP employed. However, since it is not required that all packets belonging to a flow are marked with the same DSCP label, another possible way to exploit DiffServ is to identify marking strategies for packets belonging to the same flow.

Several packet marking algorithms have been proposed for TCP flows. The marking strategy is enforced at the ingress node of a DiffServ domain (edge router). Within the DiffServ domain, marked packets are handled in an aggregated manner, and receive a different treatment based on their marked DSCP. Generally, a two-level marking scheme is adopted, where packets labelled as IN receive better treatment (lower dropping rate) than packets marked as OUT. Within the network, dropping priority mechanisms are implemented in active queue management schemes such as RIO - Random Early Discard with IN/OUT packets [1].

The basic idea of the proposed algorithms is that a suitable marking profile (e.g. a token bucket which marks IN/OUT profile packets) may provide some form of protection in the case of congestion. A large number of papers [1], [2], [3], [4], [5], [6] have thoroughly studied marking mechanisms for service differentiation, and have evaluated how the service marking parameters influence the achieved rate.

More recently, TCP marking has been proposed as a way to achieve better than best effort performance [7], [8], [9]. The idea is that packet marking can be adopted also in a scenario of homogeneous flows (i.e. all marked according to the same profile), with the goal of increasing the performance of all flows. In particular, [7], [8] consider long lived flows and adopt goodput and loss as performance metrics. Conversely, [9] focuses on WWW traffic, mostly characterized by short-lived TCP flows, and proposes a new scheme able to reduce the completion time of an http session.

In all the above mentioned marking schemes, most of the packets in the network are of type OUT. Hence, packets marked as IN will be protected against network congestion (indeed [9] relies on this property to protect flows with small window, when packet losses cannot be recovered via the fast retransmission algorithm). As shown in section II, our marking strategy is based on a somehow opposite philosophy.

Our marking strategy has been originally proposed in [10], along with a preliminary performance evaluation, devised to show that our proposed scheme consistently achieves better performance than that of an all-IN packet marking scenario, even under a changing offered traffic. In [10] we considered long-lived TCP flows, so it could appear questionable whether our scheme is able to improve performance in a more realistic scenario, where very short flows do not allow the algorithm to reach a 'steady state'. For this reason we here extend the performance evaluation to a more realistic scenario, coming from a Internet measurement campaign [14]. In addition, we compare our algorithm with one explicitely developed in order to reduce the completion time for short-lived http flows [9]. In what follows we refer to this scheme as Web Packet Marking (WPM). After the comparison we suggest the possibility to merge the two different approaches.

The rest of this paper is organized as follows. Section II describes the rationale behind the proposed marking strategy. Section III describes our adaptive packet marking (APM) algorithm, and briefly reviews the Web Packet Marking (WPM) strategy proposed in [9], used in the following sections for performance comparison. Section IV presents the simulation scenario and parameters. The performance evaluation of the proposed algorithm and the comparison is carried out in section V. In section VI we present how the APM and WPM

approaches can be joined. Finally, conclusive remarks and further research issues are given in section VII.

II. Rationale behind our Marking strategy

In [10] we proposed a new mechanism, hereafter referred to as Adaptive Packet Marking (APM), devised to increase the performance experienced by TCP flows. The fundamental difference of our algorithm with respect to the previously mentioned marking strategies is that our algorithm marks most of the packets as IN, and interleaves IN packets with occasional OUT packets. The length of an IN-packets burst is adaptively set based on an heuristic estimation of the experienced packet loss ratio.

Since the large majority of packets in the network are of type IN, by marking a packet as OUT we dramatically increase the probability that this packet is dropped. In essence, the role of the OUT packet is that of a *probe*, whose goal is to early discover whether the network is getting congested.

We remark that Random Early Discarding (RED) techniques have been designed with the same philosophy in mind. By randomly dropping packets when the queue size increases above a given threshold, RED provides early feedbacks to the TCP congestion control mechanism running at the network edge. However, RED provides only a "loose" form of control mechanism; dropped packets may belong to a subset of offered flows (which consequently reduce the emission rate), while other TCP flows may not experience packet dropping and thus remain unaware of the congestion situation. Moreover, in the same time, a few TCP flows may even experience multiple packet dropping and therefore be severely penalized.

Conversely, our proposed marking strategy provides a stronger form of control mechanism, as the packets that are likely to experience dropping are not randomly chosen among all the offered packets, but are the ones specifically marked as OUT. This operation allows to smoothly and fairly drive the TCP congestion control operation and makes the traffic offered to the network more regular, so better performance are achieved. Moreover, as we showed in [10] the higher the dropping rate of OUT packets versus the IN dropping rate, the better the performance gain is. Ideally, the optimal operational condition in the network should be that of a 100% loss rate of the OUT packets but still no loss encountered by IN packets.

When compared with the schemes proposed in [7], [8], [9], our marking strategy presents two major differences. First, the majority of packets are IN. Second, the performance takes advantage of a very high OUT packet loss rate. The fact that our strategy performs well is apparently in conflict with some results presented in [9], which show that interleaving IN and OUT packets may have a highly negative impact on the TCP throughput, if the loss rate of the OUT traffic is much larger than that of the IN traffic. In particular, a throughput reduction may be encountered as long as the percentage of IN traffic becomes greater than a given threshold. Indeed, we too have observed performance impairments for both a token-bucket marker and for a marking scheme very similar to the one proposed in [7], [8] (protection of small window and retransmitted packets, an OUT packet inserted every n IN packets[1]). However, we remark that these schemes are not designed to be adaptive to the network congestion status, while ours uses some heuristics to provide adaptability.

III. Packet Marking Algorithms

In this section we describe the proposed Adaptive Packet Marking mechanism, and we briefly review the Web Packet Marking (WPM) strategy proposed in [9]. Both packet marking algorithms can be implemented at the ingress router and act on a per-flow basis.

A. Adaptive Packet Marking

When the ingress router detects a TCP SYN packet, meaning that a new flow is offered, it reserves a state for the flow. This state is composed of the following variables:

- SN_h. This variable stores the higher Sequence Sumber (SN) transmitted by the flow. It is initially set to the ISN value (Initial Sequence Number) carried by the SYN packet, and it is updated whenever a non-empty packet (i.e. non a pure ACK packet) with higher SN arrives at the router. Higher SN is to be intended in a cyclical sense: we recall that sequence numbers wrap when the value $2^{32} - 1$ is reached.
- L_{IN}. This counter is initially set to 0. It is reset to 0 when either a packet loss is detected, or a packet marked as OUT is transmitted. It is increased for every transmitted subsequent packet. Hence, the counter L_{IN} represents the actual length of a burst of IN-marked transmitted packets.
- A_{IN}. This is an Auto Regressive filtered value which keeps track of the past values of the counter L_{IN} (i.e. the size of a burst of successfully transmitted IN packets averaged over a recent period). It is initially set equal to a design parameter A_0. In addition, as shown below, it is used by the marker to determine which packet to mark as OUT, and it is increased after every OUT-marked packet to provide adaptivity.

The algorithm is described in the flow-chart in figure 1. When a non-empty packet arrives at the router, its sequence number SN is read. According to the new SN value, and the recorded highest sequence number encountered before, we face two possible situations. If $SN \leq SN_h$, then the incoming packet is a replica of a previously transmitted packet. This means that such packet has probably been lost. Conversely, if $SN > SN_h$ the incoming packet is a new one.

Our algorithm distinguishes these two cases. In the case of packet loss, the value A_{IN} is updated as the weighted sum of the previous estimate with the current value of L_{IN}. The L_{IN} value is then reset to 0, meaning that a new burst of IN-marked packets has begun. The retransmitted packet is delivered marked as IN. In the case of a new incoming packet the current IN-marked packet burst size is increased by one. The packet is then marked as IN if the current burst L_{IN} is

[1]Neither in [7] nor in [8] the authors indicate the number of available IN tokens used in the simulations.

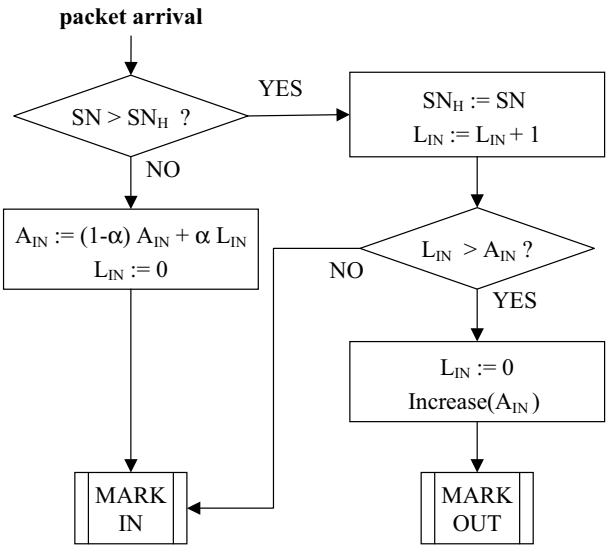

Fig. 1. Packet Marking Algorithm

shorther than the value A_{IN}. Conversely, if the actual burst of IN-marked packets has become longer than A_{IN}, the actual packet is marked as OUT, and a new burst begins ($L_{IN} = 0$).

Note that, after the transmission of an OUT packet, we need to increase the value A_{IN}. In figure 1, this operation is generically indicated as increase(A_{IN}). In fact, when congestion conditions occur, several packet losses may be encountered, and thus the value A_{IN} decreases (left part of figure 1).

To better understand how this increment should be quantitatively accounted, consider the situation in which all packets labelled as IN are successfully received, while all packets labelled as OUT are discarded. This means that the congestion level in the network has reached a given stationary target value. To remain in such stationary conditions, the OUT marking rate should not vary with time, i.e. an OUT packet should be marked every $\bar{A}_{IN}$ IN packets, being $\bar{A}_{IN}$ a constant[2] In the assumption of stop&wait TCP operation[3], no IN packet loss, and 100% OUT packet loss, it is easy to see that A_{IN} remains constant to an initial value $\bar{A}_{IN}$ if the increase rule is $A_{IN} := A_{IN}/(1-\alpha)$.

The thorough optimization of the algorithm's configuration parameters (namely, α, A_0, and the increase(A_{IN}) rule) is out of the goals of this paper, and is object of current research activity. To obtain numerical results, unless otherwise specified, we have adopted $\alpha = 0.5$, $A_0 = 7$, and $A_{IN} := 2A_{IN} + 1$ as increase rule. It is interesting to remark that even with parameters chosen without any accurate tuning, the performance of the algorithm are very good. This is perhaps an indication of the robustness of the considered algorithm to non optimal settings.

[2]It depends (in a non trivial manner) on the RIO configuration at the bottleneck link and on the number of offered flows.

[3]For general values of the contention window, such an analysis is much more complex as it further depends on how many packets have been sent when a triplicate ACK arrives at the sender.

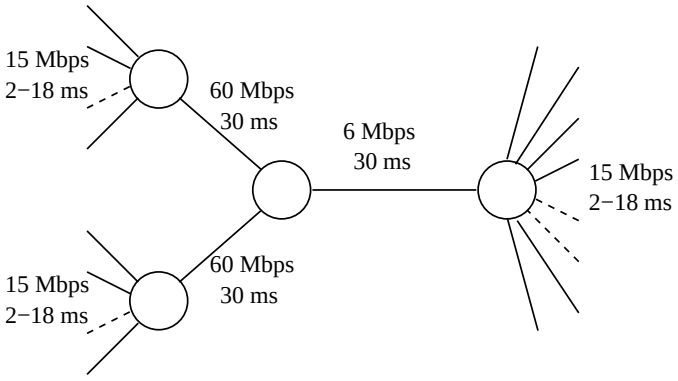

Fig. 2. Network topology

B. Web Packet Marking

In [9] the authors present two packet marking schemes. The first one is tightly integrated with the TCP protocol: the source is allowed to send up to N_s IN packets when it starts, and then up to $N_a = sstresh$ at the beginning of a Slow Start phase, and up to $N_a = cwnd$ at the beginning of a Fast Recovery phase. The second scheme does not require the knowledge of internal TCP variables, but it uses a constant value $N_a = N_s = 5$, hence this scheme can be implemented at ingress router. In what follows we compare the second scheme with our adaptive scheme.

IV. Simulation Scenario

The network topology considered is shown in figure 2. It consists of a single bottleneck link, whose capacity is set equal to 6 Mbps. Propagation delays on the access links are chosen so that Round Trip Times are different (from 124ms to 198ms, the average value is 160ms).

To simulate WWW-like traffic, a number of TCP-Reno sources are connected to each ingress router. The flow's arrival rate is modeled as a Poisson process and the flow lengths are drawn from the distribution given in table I. This distribution has been constructed from data collected at the end of 2002 through the Tstat tool [14] on the Internet access link of Politecnico di Torino, i.e. between the border router of Politecnico and the ingress router of GARR, the Italian research network. Collected flow length data have been ordered from the shortest to the longest and divided in 15 groups, each corresponding to a 0.066% probability. Table I reports, for each group, the average flow length measured both in bytes and in IP packets (for simplicity, we have considered 1500 bytes packets).

In our simulations we have considered two different WWW offered loads, corresponding to 64% and 90% of the bottleneck link capacity. Since results are very similar, in what follows only 64% load results are presented. One continuous backlogged TCP flow has been added.

Each router is equipped with RIO (RED with In/Out bit, [1]) as Active Queue Management. For RED operation refer to [11]. We let min, max be the two thresholds, w_q the weight of the instantaneous queue value in the moving average filter, P_{max} the maximum dropping probability in the region

TABLE I

FLOW LENGHT DISTRIBUTION

Group	Bytes	Packets	Group	Bytes	Packets
1	119	1	9	1650	2
2	179	1	10	2861	2
3	251	1	11	4706	4
4	334	1	12	8015	6
5	428	1	13	13681	10
6	529	1	14	26641	18
7	658	1	15	284454	190
8	948	1			

of random discard. RIO uses two twin RED algorithms for dropping packets, one for IN packets and one for OUT packets which share the same physical queue. So RIO is configured with two sets of RED parameters: (min_{in},max_{in},$P_{max_{in}}$) and (min_{out},max_{out},$P_{max_{out}}$). RIO discriminates against OUT packets in times of congestion essentially in two way: firstly IN dropping probability depends on the average queue for the IN packets, while OUT dropping probability on the average total queue; secondly parameter are opportunely chosen for the two kinds of traffic. In [1] the authors suggest the following rules: $min_{out} < min_{in}$, $max_{out} << max_{in}$, $P_{max_{out}} > P_{max_{in}}$, and in the paper they choose $max_{out} < min_{in}$.

As regards RED parameters, the thresholds and $P_{max_{in}}$ are chosen according to [12], i.e. $max = 3min$, $P_{max_{in}} = 0.1$. Following [13], the filter coefficient w_q is set to $w_q = 1 - exp(-M/(C * 10 * RTT)) = 0.0012$, where C is the link capacity, M is the packet size and RTT is the Round Trip Time.

RIO configuration allows the network provider to trade off link utilization and delay performance: the higher the RED thresholds, the higher link utilization and delay. Different kind of settings have been considered for APM, WPM and the "no marker" situation (NM in what follows). In [10] we showed that APM performance is better as the service differentiation among OUT and IN packets increases. So here we let the min_{in} threshold values going from 9 to 240 packets, while the OUT traffic settings are fixed to $min_{out} = 2$, $max_{out} = 6$ and $P_{max_{out}} = 0.2$. For the WPM algorithm, where the majority of the packets are marked OUT, we let the min_{out} threshold values going from 9 to 240 packets, while the min_{in} threshold is so high that no IN packets are dropped. When no marker is applied, all the packets are considered IN and RIO configuration is the APM one.

Lastly queue physical lengths were chosen so that packet losses occurred only in the core router, due to RIO (not to physical queue overflow).

We considered two main performance figures: the average packet delay and the average flow completion time, i.e. the time from the emission of the SYN packet to the reception of the last data packet (no connection closing has been simulated). These values are plotted versus the average goodput (at the application level) for each of the threshold setting, so we can obtain the "performance frontiers".

Simulations were conducted through ns v2.1b9a. We used TCP Reno implementation. For each configuration at least

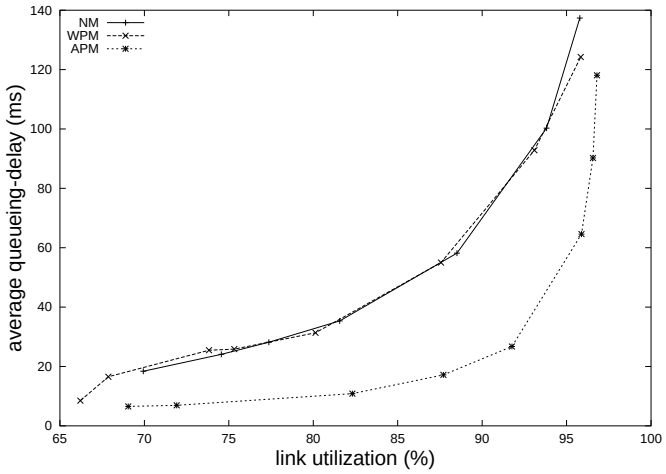

Fig. 3. Delay vs Goodput

5 simulations with different random seeds were run. Each simulation lasted 500 simulated seconds, statistics were collected for 500 simulated seconds after discarding the initial 50 seconds. In the figures we present in the following section, standard deviation of goodput is always less than 2% and standard deviation of delay and completion time is always less than 5% of their numerical value.

V. PERFORMANCE EVALUATION

As said in the previous section, throughput and delay performance depend on the RED threshold settings configured by the network administrator. To provide a wider performance evaluation, rather than selecting a given RED configuration, we have run results for several different configurations.

Results shown in Figure 3 are reported in terms of the so called "performance frontier", i.e. delay vs goodput performance. APM does not act on very short flows (less than 7 packets), so the remarkable advantage in figure 3 is achieved by controlling the longest flows (groups 13, 14, 15 and the long-lived TCP flow), whose throughput is regulated by the insertion of OUT probe packets. Instead, we note that the performance of WPM are very similar to that experienced in the case of no marking (NM) algorithm employed (all packets marked as IN). This is expected, as WPM is not specifically designed to improve goodput/delay performance, but it is designed to reduce the completion time for short flows.

Completion time results are shown in figures 4, for the three cases of i) single packet flows; ii) 18 packet flows, and iii) 190 packet flows. Let us first focus on the cases of 1 and 18 packet flows. As expected, WPM is effective (and slightly better than APM) when the link utilization is low, essentially because it protects the first packets in the flow, whose loss can be recovered only via retransmission timeout expiration (and not via fast retransmit). At high utilization, this "protection" effect reduces, as packet loss percentage decreases (results in table II), and queueing delay becomes the main contribute to completion time. For this reason APM provides results consistently better than NM and WPM, because it is the only

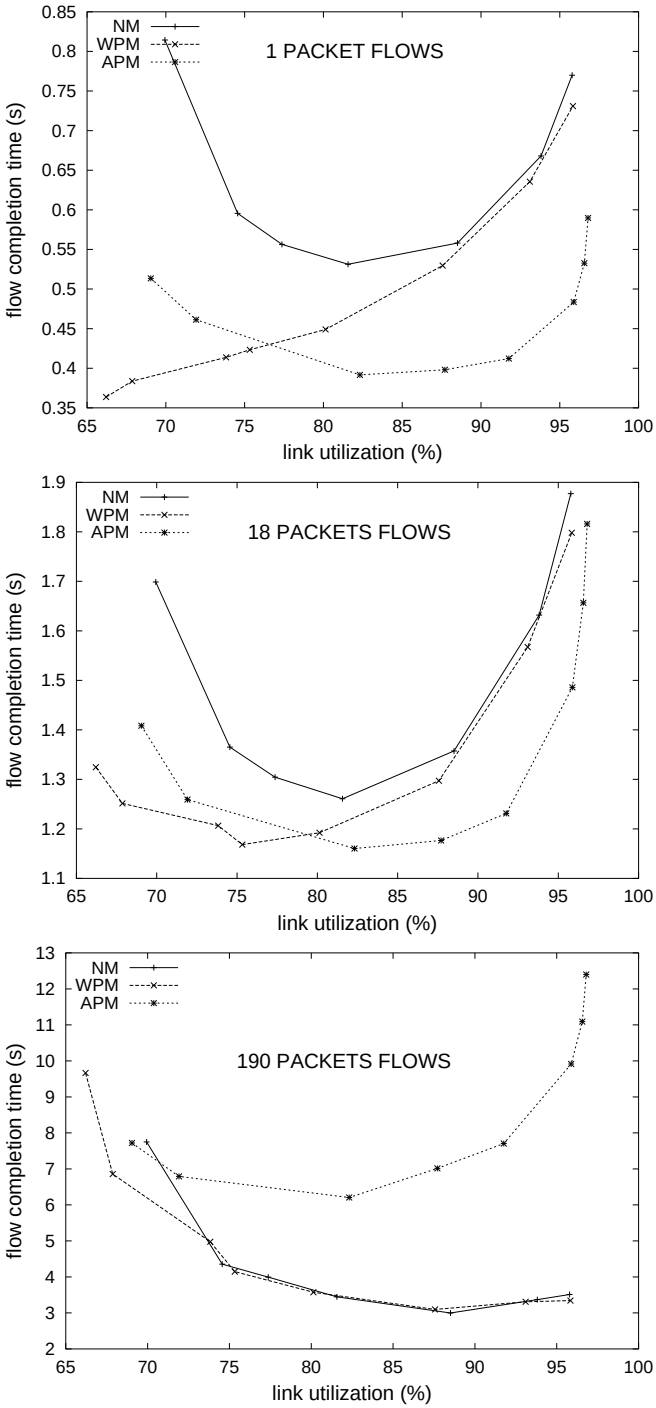

Fig. 4. Flow Completion Time vs GoodPut

marking strategy that allows to keep the queue occupancy low in high utilization conditions [4].

The completion time results for the case of 190 packet flows (third plot of figure 4) shows that, in high utilization conditions, APM "pays" the improved completion time for short-lived flows with a significantly higher completion time for long-lived flows. The reason is that, in high utilization conditions, long-lived flows are given sufficient time to "adapt" (i.e. increase the number of OUT-marked packets) to the congestion situation, while short-lived flows behave in a more aggressive manner.

[4]We remark that these insights on the performance of the considered algorithms were made possible only by the choice of presenting results in terms of performance frontiers rather than selecting a specific RED configuration. Indeed, several contrasting results presented in the TCP literature are motivated by different behaviors in different operational conditions - selecting a single RED configuration allows to achieve performance for just a single operational condition.

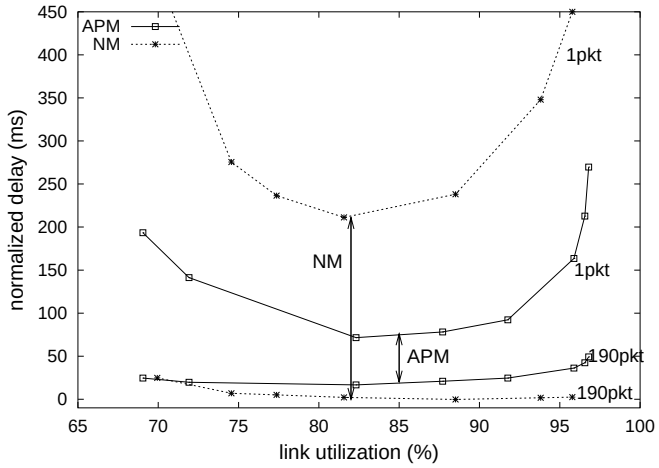

Fig. 5. Normalized per-packet delay

Despite this significant worsening in comparison to NM and WPM, we remark that this is not necessarily an impairment. To prove this point, figure 5 compares the normalized per-packet delay of APM with that of NM, plotted versus the link utilization, for 1 packet flows and 190 packet flows. This normalized delay is evaluated as the difference between the average completion time and the minimum completion time (when the network is unloaded), divided by the total number of data packets. From figure 5 it appears that, with APM, this normalized delay is less sensitive to the network configuration (as it is the average per-packet delay in figure 3). More importantly, APM is able to reduce the variability for different flow lengths in comparison to the no-marker scenario. So APM control on longest flows can be seen as a way to reduce the common TCP unfairness between short and long flows, caused by the fact that long flows can rely on the Fast Retransmit and Fast Recovery algorithms and have better Round Trip Time estimates. This unfairness holds also for WPM traffic for high link utilization: WPM normalized-delay curves are very similar to NM ones.

Finally, the difference among NM, WPM and APM in terms of percentage of IN packets, and percentage of dropped packets appears evident from table II, where their range of variation for the different RIO settings is shown (from lower thresholds to higher one).

VI. A NEW THREE COLOR MARKER

From previous results it appears clear that our scheme and WPM are in some way orthogonal, so we can think to merge them. In order to elaborate a new scheme we need three different kind of packets. In a DiffServ framework, the

TABLE II

IN PACKETS AND DROPPED PACKETS PERCENTAGE

	NM	WPM	APM
IN%	100	26-13	97-98
DROP% TOT	5.3-0.0	4.5-0.0	2.9-2.0
DROP% IN	5.3-0.0	0.0-0.0	1.1-0.0
DROP% OUT	–	6.2-0.0	64-98

domain administrator can dedicate an Assured Forwarding (AF - see RFC 2597) class i with three different dropping level to marked TCP traffic: AFi0, AFi1 and AFi2 ordered for increasing dropping probability in core routers. Vulnerable packets according to the WPM scheme can be marked as AFi0, probing packets according to APM can be marked as AFi2, while the majority of packets are marked as AFi1. Performance evaluation shows that this new scheme, called Merge Packet Marking (MPM) is able to sensibly improve APM marking behavior in low link utilization without any other drawbacks. For example figure 6 shows the average completion time for 1 packet flow when WPM, APM and MPM are employed.

VII. CONCLUSIONS

In this paper we have presented some results of an extensive performance evaluation for an adaptive packet marking (APM) scheme in an heterogeneous scenario with different TCP flow lengths. Numerical results show that APM provides excellent performance in high utilization conditions, and protects short-lived flows from performance impairments due to packet losses and consequent recovery via retransmission timeout expiration. Moreover, in medium utilization conditions, APM provides improved performance in comparison to a no-marker scenario, but also comparable with those of another algorithm, called Web Packet Marking (WPM), whose target is explicitely WWW traffic. These two schemes exhibit improvements for different network configuration (low and high utilization), so we have proposed a new three colors marking scheme, called Merge Packet Marking (MPM), which combines the APM operation with that of WPM.

Since the APM approach is based on some heuristics, we think that further improvements of the marking mechanism are possible. In particular, current research activity is investigating the adaptation law of the parameter A_{IN}, and the effect of different initial settings.

ACKNOWLEDGMENT

The authors would like to thank Dr. Marco Mellia for the helpful discussions and for the 2002 data traffic in table I.

REFERENCES

[1] D. D.Clark and W. Fang, “Explicit Allocation of Best Effort Packet Delivery Service”, IEEE Transactions on Networking, Vol. 6, No. 4, pp. 362-373, Aug. 1998.

[2] J. Ibanez, K. Nichols, “Preliminary Simulation Evaluation of an Assured Service”, IETF draft, August 1998

[3] N. Seddigh, B. Nandy, P. Piedu, “Bandwith Assurance Issues for TCP flows in a Differentiated Services Network”, IEEE Globecom, Rio de Janeiro, pp. 1792-1798, December 1999

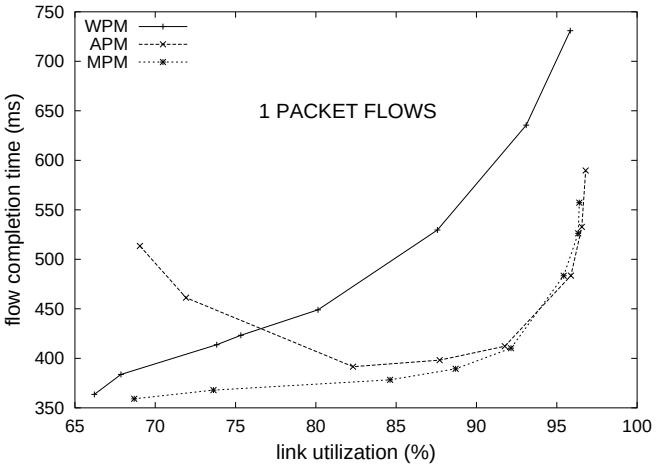

Fig. 6. Flow Completion Time vs GoodPut

[4] S. Sahu, D. Towsley, J. Kurose, “Quantitative Study of Differentiated Services for the Internet”, IEEE Globecom, Rio de Janeiro, pp. 1808-1817, December 1999

[5] S. Sahu, P. Nain, D. Towsley, C. Diot, V. Firoiu, “On Achievable Service Differentiation with Token Bucket Marking for TCP”, Proc. ACM SIGMETRICS’00, Santa Clara, CA, June 2000

[6] W. Feng, D. Kandlur, D. Saha, K. Shin, “Adaptive Packet Marking for Mantaining End-to-End Throughput in a Differentiated Services Internet”, IEEE/ACM Transactions on Networking, Vol. 7, NO:5, pp. 685-697, April 1999

[7] F. Azeem, A. Rao, S. Kalyanaraman “A TCP-Friendly Traffic Marker for IP Differentiated Services” IwQoS’2000, Pittsburg, PA, June 2000.

[8] G. Lo Monaco, F. Azeem, S. Kalyanaraman, Y.Xia, “TCP-Friendly Marking for Scalable Best-Effort Services on the Internet”, Computer Communication Review (CCR), Volume 31, Number 5, October 2001.

[9] M. Mellia, I. Stoica, H. Zhang, “Packet Marking for Web traffic in Networks with RIO Routers”, Globecom 2001, San Antonio, Texas, November 25-29, 2001

[10] G. Neglia, G. Bianchi, F. Saitta, D. Lombardo, “Adaptive Low Priority Packet Marking for better TCP performance”, Net-con 2002, Paris, October 2002

[11] S. Floyd, V. Jacobson, “Random Early Detection gateways for Congestion Avoidance” IEEE/ACM Transactions on Networking V.1 N.4, August 1993, p. 397-413

[12] S. Floyd, “RED: Discussions of Setting Parameters”, email November 1997, http://www.icir.org/floyd/REDparameters.txt

[13] S. Floyd, R. Gummadi, and S. Shenker, “Adaptive RED:An Algorithm for Increasing the Robustness of RED’s Active Queue Management”, August 1, 2001, under submission

[14] M. Mellia, A. Carpani, R. Lo Cigno, “Measuring IP and TCP behavior on Ege Nodes”, Globecom 2002, Taipei, TW, Novemer 2002