

Fair Marking of Web Flows using Partial State Information*

M. Alzati, M. Bottigliengo, C. Casetti, M. Mellia

Dipartimento di Elettronica, Politecnico di Torino, Torino, Italy

In this work, we propose a class of Fair Markers that, without the penalty of per-flow management, achieves the same performance enhancement of a per-flow marker, but with a much simpler design and limited amount of memory. Simulations with realistic traffic scenarios allow us to compare our proposal with existing ones, and an analytical model is also proposed to predict the performance of TCP flows.

1. Introduction

One of the most frequently debated Internet QoS architecture, DiffServ [1] handles flow aggregates, performing packet classification into coarse classes at the network ingress and supporting different per-class guarantees at every hop in the network core.

When the flow aggregate conforms to a *Committed Information Rate* (CIR) (with the possible addition of a temporary, limited surplus of bandwidth), then packets are marked as *in-profile*. Otherwise, when the flow aggregate is non-conformant to the CIR, packets are marked as *out-of-profile*. Once packets have been classified at edge network nodes, they will be treated by core nodes using differentiated dropping probabilities, so the in-profile packets will be discarded only after all out-profile traffic has already been dropped. At network edge, packet markers such as the *Time Sliding Window Three Color Marker* [7] (TSWTCM) must be used to mark packets². This marker considers the packet flows belonging to an SLA (Service Level Agreement, defining User QoS requirements and network allocation of resources) as pure aggregate of packets, and does not distinguish them based on higher-layer protocols.

A common problem of packet markers, underlined in previous works, is their inability to provide a similar level of protection to long- and short-lived TCP flows alike. In particular, short-lived flows, representing the majority of today's Web traffic, suffer from packet losses occurring during or just past the three-way handshake phase, when the TCP congestion window size may not be large enough so as to trigger the Fast Recovery algorithm. Such flows are delayed by (possibly) repeated timeouts that are affecting them only because of their state, while longer flows manage to avoid timeouts thanks to the reception of duplicate ACKs that enable Fast Recovery. A similar situation occurs right after a timeout, when a flow is more vulnerable because of its small window, and the danger of repeated timeouts caused by erroneous packet marking is consistent. This gives rise to unfairness issues, where long-lived flows manage to obtain the largest benefits from the adoption of a DiffServ scenario.

*Part of this work was supported by a contract with the NEC Network Laboratories of Heidelberg, Germany

²In this paper we will refer to in-profile traffic as *Green*, while *Yellow* class will be used to represent both out-of-profile and Best Effort traffic

The main contribution of this paper is twofold: the proposal a Fair Marker that, by using a limited amount of memory in edge nodes, achieves the same performance enhancement of a per-flow marker, but with a much simpler design; also, a simple analytical model of the completion time of short-lived flows in a DiffServ domain using our proposed Fair Marker.

2. Fair Marker Design

Previous researches have pointed out that the DiffServ architecture cannot fully guarantee QoS requirements and fairness to TCP flows, and, in particular, to Web-like file transfers. In particular, as already pointed out in [4–6], a direct consequence of using a simple TSWTCM, is the impossibility of discriminating packets belonging to "TCP fragile flows", i.e., flows whose TCP congestion window is small. The reason is that TCP suffers when ACKs are not enough to trigger Fast Recovery and Fast Retransmit, so that RTOs are experienced. It is therefore important to provide a minimum protection to client traffic (both data and signalling) so as to better exploit DS capabilities.

The design guidelines of the Fair Markers follow those stated in previous works; they can be summarized by the following simple principles:

- detect when a TCP connection is in Slow Start, and mark as *Green* the first few segments of the connection;
- detect when a TCP connection is performing a retransmission, and mark as *Green* the retransmitted segment as well as the first few segments following it.

We will refer to the situation just outlined as "critical states". The practical effect of marking the first packets of a flow is to notify the network that it is too soon to signal the new flow about ongoing congestion: the benefit to the network would be marginal, what with the small window size of that connection, while the flow would be excessively penalized from the onset.

Of course, the simple inspection of a TCP segment does not allow the marker to infer for how long the flow has been active, or whether or not it is in one of the critical states, therefore some form of connection tracking is required. To account for scalability, the solution we chose is to limit the number of flows for which state is kept, trying to identify a small set of flows that have been most recently active. In Figure 1, the general architecture of the Fair Marker is sketched: the operations of each block are listed below, with the understanding that further details can be found in [8]:

- **Flow Classifier:** has the function to update the list of active flows, within memory constraints. The core of the Flow Classifier is a finite-size data structure called *Short-Range Flow Table* (SRFT), that supplies a snapshot of the most recently active flows across a node. The table size, i.e., the maximum number of tracked flows, is limited to `SRFT_size`. Its structure is a list collecting the most recent flow IDs that have sent at least a packet. It also provides a bookkeeping of the amount of data sent by stored flows.
- **Marker:** passes the packets of a flow to the Policers pre-marking them according to a threshold t_{ft} (possibly adjusted by the Policers);
- **Meter:** estimates the rate of the packets at the ingress of the Policers by using a moving-window average;
- **Policer:** possibly re-marks the packets by comparing the average rate with the CIR, in order to keep the rate of the ingress packets within the SLA.

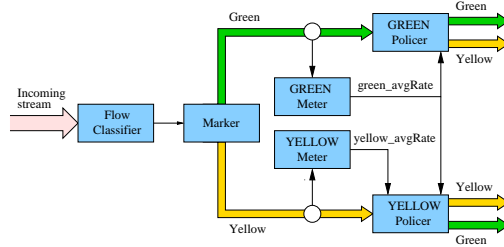


Figure 1. Architecture of the Fair Marker

To perform the active management of the queues in edge routers, we have chosen MRED. Only one physical queue is implemented in each ingress of the router, and this queue is split into two virtual queues: in order of priority, one for segments marked as *Green*, and one for *Yellow* segments and Best Effort (unmarked) ones. With this choice, only one average queue dimension is used.

Each physical queue is managed according to the *FIFO (First In First Out)* scheme, therefore the sharing within virtual queue is only meaningful to MRED. All simulations were performed using a staggered parameter set for MRED: maximum queue size of 200 segments; min_{th} , max_{th} and max_p for *Green* traffic equal to 60 segments, 140 segments and 0.1; min_{th} , max_{th} and max_p for *Yellow* and BE traffic equal to 5 segments, 60 segments and 0.2. The reason behind the choice of a staggered parameter set rather than an overlapped one lies in its property of lower drop preference protection: *Green* marked segments drop begins after all *Yellow* segments have been dropped. Since we were not really concerned with RED performance, the parameters we chose reflect those suggested in the literature [10].

3. Performance evaluation

A realistic traffic model is used to investigate the impact of DS over Web-like traffic that crosses both forward and backward congested domains. It is thus necessary to first outline the simulation settings and the traffic model that was used throughout this work.

3.1. Network Scenario

Simulations were run under *ns-2.1b8a* and aim at evaluating the performance of a network of *three* pairs of client-server clouds, showed in Figure 2. The pairings identify client-server relationships, their connection data being carried over a 10-Mbit/s bottleneck link, characterized by a 20 ms delay. At each end of the bottleneck link, two edge routers mimic ingress/egress nodes (depending on the direction of traffic flows) of a DiffServ domain: they are tagged R1 and R2 in the Figure. Each receives traffic from the attached clouds in the form of aggregate TCP connections and marks these segments according to the specific SLA; then, following the DS guidelines, it places packets in one virtual queue. Router queues use *MRED (Multi-level Random Early Detection)* active management techniques.

In the scenario in Figure 2, two client clouds are attached to R1 and one client cloud is attached to R2; server clouds are correspondingly attached to the opposite edge node. Two client clouds connected to R1 send requests to servers connected to R2. One cloud will always send Best Effort (BE) traffic, while the other one will be configured so that i) servers and clients uses the BE class, ii) servers manage to send DS traffic, while clients send only BE packets, or iii) both servers and clients use a portion of DS-reserved bandwidth.

Table 1
 Lengths of Server Responses (avg: 13666, 14.2 PKT)

Flow lengths in bytes	P2S	Flow lengths in bytes	P2S
61	1 PKT	4149	5 PKT
239	1 PKT	6358	7 PKT
539	1 PKT	10910	11 PKT
1349	2 PKT	19878	20 PKT
2739	3 PKT	90439	91 PKT

A second BE cloud pair (with server connected to R1 and clients to R2) is added to simulate possible congestion on the reverse path, i.e., where request segments of the other two clouds arrive: therefore, server data of the third cloud pair are routed over the same path as client requests from the other two pairs. Each cloud includes one router with multiplexing/marketing functionalities, to which all active sources are connected, managing traffic either coming from servers or coming from clients. These routers are linked to the edge routers by a 1-Gbit/s link, with a 1- μ s latency, so they are never congested. In order to evaluate the classic DS behavior, the marking strategy we selected is the standard TSWTCM [7], used in two-color mode (i.e., by setting PIR=CIR): therefore, packets are only marked as either *Green* or *Yellow*, and not *Red*; in this paper we will refer to it as TSW2CM.

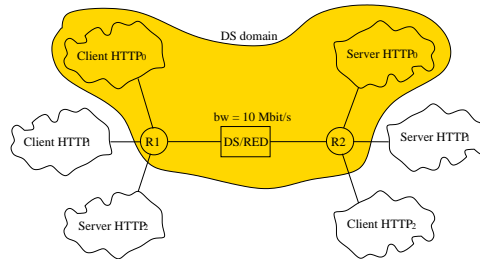


Figure 2. Network Scenario Used in Simulation

3.2. Traffic Model Description

The traffic model was designed to mimic HTTP flows as realistically as possible, comprising requests from clients and responses from the servers, each connection being simulated from the three-way handshake until its termination, using the Reno Full-TCP *ns* agent. Flow scheduling times and request/response sizes are chosen as close as possible to an actual client/server Web-like transaction.

In order to realistically simulate the size distribution of HTTP flows, we define two distributions of possible flow lengths, one for client requests and another for server replies. In each distribution, the flow lengths are indicated as number of packets to send (P2S), which can be computed from the size in bytes when the segment size is set. For the sake of simplicity, each segment across the network is full-sized at 1000 bytes: therefore the P2S value, obtained dividing the size in bytes by the segment size, must be approximated in excess.

Following the methodology already used in [4,5], in the simulations reported in the following Sections, client requests consist of a number of segments which can vary from one to three:

one-segment requests account for 85% of the total, two-segment requests account for 10%, while the remaining 5% requests consist of three segments. Many requests have only one segment because today the greatest number of HTTP interactions between client and server consist of HTML pages requests, which need only a small number of bytes. The flows with two or three segments simulate particular requests which need more bytes, for example, requests of active pages where the user submits client-side information.

Once a request has been completely received from a server, the size of each reply is set choosing in a randomly uniform fashion among the flow lengths listed in Table 1, that was derived in [4,5] to best match real traffic distribution.

At the client side, each DiffServ cloud generates web request flows that follows a Poisson process. Requests are separated by a random delay, which is chosen using an exponential distribution, with average value computed considering B as the bandwidth of the bottleneck link, expressed in Mbit/s; ρ as the ratio of the total offered load to the bottleneck bandwidth; α_i as the fraction of global traffic generated by cloud i , $\sum_i \alpha_i = 1$.

It is straightforward to observe that, with the flow lengths listed above, the more heavily loaded direction of the bottleneck link is the one from server to client (with an average of 14.2 packets per flow, instead of only 1.2 packets per flow in the opposite direction). Consequently, the level of traffic chosen in our simulations refers only to server replies and not to client requests.

In this paper the network was tested by fixing the total server load (ρ) at 80% of the bottleneck bandwidth, i.e., at 8 Mbit/s, and the DS load (α_{DS}) was varied between 60% and 95% of the overall traffic. The CIR on the server-to-client path is set to 6.4 Mbit/s, i.e., 80% of the total offered load. This allows us to identify two operating regions, one where the DS load is below the CIR (SLA overprovisioning), and one where the DS load is above the CIR (SLA underprovisioning). A vertical dotted line separating the two regions is reported on plots where the DS load was varied.

Considering the client-to-server path for DS flows, two scenarios can be tested: one where no explicit reservation for client traffic is performed, and one where client packets are allocated a 560 Kbit/s CIR, which in our traffic model corresponds to about 80% of the client offered load. In this paper, we report results for the latter, while the no-reservation case was thoroughly studied in [8].

In all scenarios, the third pair (pair 2), the one that generates traffic to load the edge routers on the reverse path, always activates BE connections so that the traffic it generates reaches an average rate of 8 Mbit/s.

We also ran simulations where we vary the total offered load ρ , maintaining the DS load at a fixed rate. Again, these results are not included in this work, but they can be found on [8]. They largely confirm the same findings we include in this paper.

Finally, simulation of bursty, short-lived flows need long runs in order to be confident with the obtained results. In our study, to get accurate results, each simulation is run for at least 600s, so that the initial transient phase has little impact on the results.

We select as main QoS performance index the *Completion Time* (CT) of an http transaction, that takes into account the amount of time required to successfully receive a server reply.

3.3. Simulation Results

Using the simulation scenario in which both clients and servers exploit DiffServ-reserved bandwidth, we have compared the TSW2CM with the FM scheme proposed above. The tunable parameters of FM were chosen as follows: $t_{ft} = 7$ and `SRFT_size`= 30. Some results justifying the choice can be found in [8].

The left plot of Figure 3 depicts the average completion time of http transactions, comparing

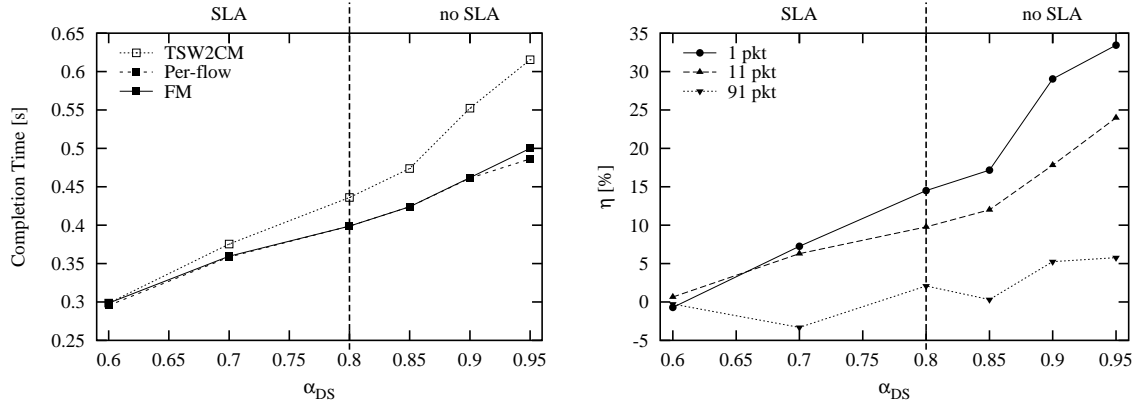


Figure 3. Completion times of http transactions using different markers: absolute (left) and relative (right) values

FM (solid line), TSW2CM (small-dot line) and a marker that maintains per-flow state³ and behaves similarly to [5,6]. As can be seen, the FM outperforms the TSW2CM, while exhibiting a negligible performance loss with respect to the per-flow marker. Also notice that the largest improvement is achieved when the offered load exceeds the subscribed SLA, in which the effectiveness of the FM becomes even more noticeable. In order to gain greater insight about the performance of flows of different size in comparison to TSW2CM, the right plot of Figure 3 shows the relative gain

$$\eta = 100 \cdot \frac{CT(TSW2CM) - CT(FM)}{CT(TSW2CM)}$$

for flows whose server reply sizes are 1-, 11- or 91-packet long. It is important to observe that the goal of improving the level of protection of shorter flows, hence the fairness, is achieved, as shown by the gain of 1- and 11-packet long flows. Furthermore, the impact on longer flows is negligible (the gain η is close to 0). The improvements shown in the plots are due to the FM ability of selecting packets belonging to flows in critical states and offering them the protection they need in order to speed up the completion of their transfers.

4. Modeling of TCP flows in DiffServ Networks

In this Section, we briefly describe a TCP model that can be used to predict the performance obtained by a flow of a given size that is carried by a DiffServ capable network. The model we are proposing is an extension of [11,12], which have been extended and simplified to deal with the DiffServ framework, and in particular to deal with TSW2CM and FM markers. Indeed, when we try to apply this model to DiffServ, we have to introduce a second average dropping probability: the first referring to *Green* segments and the second to *Yellow* or *BE* ones. We make the same assumptions as in [11]. And we further simplify the model by neglecting the Fast Recovery algorithm for flows shorter than 10 segments.

To describe our model we use the following variables, which are supposed to be given:

³The per-flow marker was actually designed as an FM with infinite SRFT size, so that flow state is never lost.

R average RTT;
 p_{tr} probability of *Yellow* segment to be remarked as *Green*;
 p *Green* segment dropping probability;
 q *Yellow* segment dropping probability;
 k *composed Yellow* segment dropping probability;
 T estimated RTO duration;
 t_{ft} SRFT threshold;

where k can be obtained simply by adding the probability of dropping a *Yellow* segment not remarked and the probability of dropping a *Yellow* segment remarked as *Green* : $k = q(1 - p_{tr}) + pp_{tr}$.

Firstly, we address the problem of modeling short-lived TCP flows. Let us evaluate the average TCP connection setup: when FM is considered and t_{ft} is at least 1, the average connection setup time C_s can be expressed as

$$C_s^1 = R + (1 - p) \sum_{i=1}^{\infty} p^i \sum_{j=1}^i 2^{j-1} T = R + T \frac{p}{1 - 2p}, \text{ if } t_{ft} \geq 1 \quad (1)$$

Note that the second sum of j may go to infinity, but in realistic TCP protocols the scaling factor 2^{j-1} is typically bounded to a maximum of 64. This simplification may not make a difference for small loss probabilities, but let us to obtain a simple expression for C_s^1 . The same expression, but with k in place of p is valid when t_{ft} is 0, and models the connection setup in a TSW2CM.

Consider now a data transfer of a flow of a given size. Let $C_{m,c}^{w,j}$ be the average time spent to successfully send m segments with an initial congestion window of size w , with c credit left to deterministically mark consecutive *Green* segments, but with j tokens available in case of RTO. Similarly, let $C_{m,c}^1$ be the time needed to successfully open a connection and send m data segments with $t_{ft} = c$.

For example, a 7-segment flow Completion Time, with t_{ft} equal to 5, is given by⁴

$$C_{7,5}^1 = C_s^1 + C_{1,1}^{1,1} + C_{6,2}^{2,5}$$

Next we derive $C_{w,c}^{m,j}$. If $m = 1$, the time needed to successfully send a packet is similar to C_s , but there are three different cases, depending on t_{ft} value:

$$\begin{aligned}
 C_{1,1}^{1,1} &= R + (1 - p) \sum_{i=1}^{\infty} p^i \sum_{j=1}^i 2^{j-1} T = R + T \frac{p}{1 - 2p} \\
 C_{1,0}^{1,1} &\approx R + \frac{k}{p} (1 - p) \sum_{i=1}^{\infty} p^i \sum_{j=1}^i 2^{j-1} T = R + T \frac{k}{1 - 2p} \\
 C_{1,0}^{1,0} &= R + (1 - k) \sum_{i=1}^{\infty} k^i \sum_{j=1}^i 2^{j-1} T = R + T \frac{k}{1 - 2k}
 \end{aligned} \quad (2)$$

With a burst of two segments we have six cases, which can be expressed by:

$$C_{2,c}^{2,j} = R(1 - P)(1 - Q) + (1 - P)Q(R + T + C_{1,j}^{1,j}) + P(1 - Q)(T + C_{1,j}^{1,j}) + PQ(T + C_{2,j}^{1,j}) \quad (3)$$

being $P = \begin{cases} p & \text{if } c > 0 \\ k & \text{if } c \leq 0 \end{cases}$ and $Q = \begin{cases} p & \text{if } c > 1 \\ k & \text{if } c \leq 1 \end{cases}$

⁴Recall that two *Green* tokens, if available, are spent during the three-way handshake.

The derivation of $C_{m,c}^{w,j}$ for other combinations of m, w can be obtained extending the previous reasoning. The reader interested in the whole derivation can refer to [13]. This procedure can be repeated to evaluate $C_{m,c}^{w,j}$ for $m > 9$, but the complexity grows exponentially due to all the possible combinations that must be taken into account. Therefore we prefer to follow the ideas in [12], and complete this model with a steady-state model to estimate the TCP performance of long lived flows.

Next, we concern ourselves with the modeling of long-lived TCP flows. To evaluate Completion Times for long lived flows we can refer to [12], in which a new model for TCP performance is presented that extends the steady-state results from [2] by deriving new models for two instances that can dominate TCP latency: the connection establishment and the initial Slow Start. We further extend that model to be able to deal with different dropping probabilities, i.e., considering yellow and green packets. In particular, we assume that $p \approx 0$, i.e., no *Green* segments are dropped; this assumption is correct only if the DiffServ network is carefully dimensioned, i.e., the provisioning of the DS bandwidth is well configured [4].

Following the same steps as in [2,12], the average throughput B experienced by a long-lived TCP source is given by:

$$B \approx \min \left(\frac{W_{\max}}{R}, \frac{Q + \frac{1}{k} + Qt_{ft} + \sqrt{\frac{8}{3} \left(\frac{1-k}{k} + Qt_{ft} \right)}}{R \left(1 + \sqrt{\frac{2}{3} \left(\frac{1-k}{k} + Qt_{ft} \right)} \right) + T} \right) \quad (4)$$

in which W_{\max} is the maximum sender congestion window, $Q = \min(1, 3/w)$ is the probability of detecting a loss by a RTO.

Assuming that the flow length distribution is known, the predicted Completion Time can be used to evaluate the average number of contemporary active flows on a bottleneck queue. Indeed, considering that a bottleneck queue crossed by TCP flows can be modeled by a M/G/1 processor sharing queue, and being able to know the service time required to serve a client, i.e., the average Completion Time for each flow length, $E[C_i]$, it is therefore simple to compute the average Completion Time $E[CT]$ as

$$E[CT] = \sum_{i=1}^{\infty} E[C_i] p_i \quad (5)$$

where i represents the flow length and p_i the probability that one starting flow consists of i segments.

Then, stating with $E[\lambda]$ the average rate at which flows arrive at the queue, from Little's formula, we obtain

$$E[N] = E[\lambda] E[CT] \quad (6)$$

Equation (6) can be used to dimension the *SRFT*, e.g, setting `SRFT_size`= $E[n]$. Figure 4 shows the comparison between the number of contemporarily active flows estimated by simulation and model, as a function of the normalized total offered load.

5. Model validation

To validate the model against simulation results, we ran several test cases, which are not reported here due to lack of space and are available in [13]. We report here only one plot, depicted in Figure 5, which shows the comparison between the proposed model and the simulation.

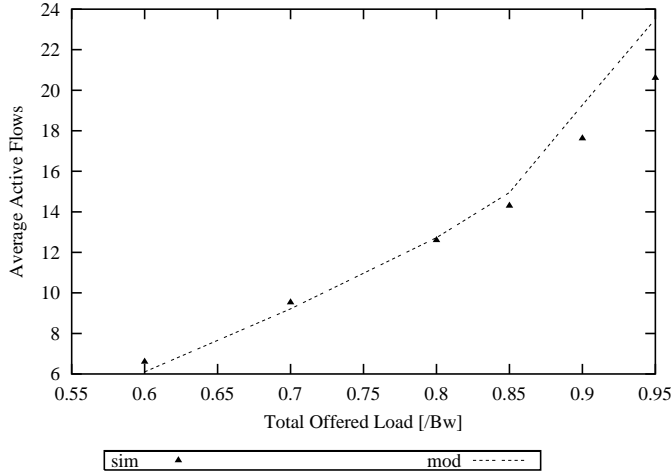


Figure 4. Number of contemporary active flows measured during simulation and obtained by the model, for different values of the total offered load.

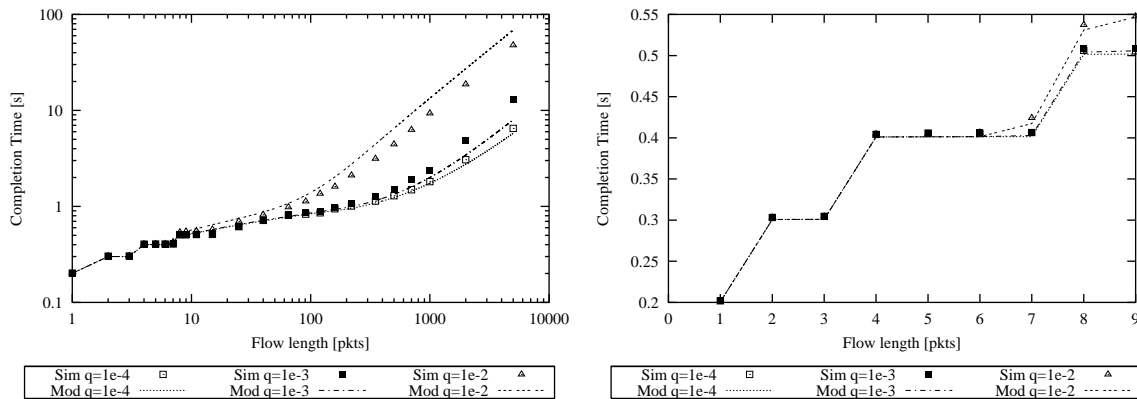


Figure 5. Completion Time for different values of the yellow dropping probability.

The simulation scenario is composed by flows that have to send a given amount of packets, and cross a buffer which artificially drop packets according to two fixed dropping probabilities. In particular, we set $p = 10^{-4}$ and $q = p$, $q = 10p$ and $q = 100p$. An FM is present, in which the t_{ft} threshold has been set to 8. It can be seen that the TCP model very well predicts the Completion Times of short-lived flows, as demonstrated by the rightmost plot, while the long-lived part of the model is less accurate.

Figure 6 instead reports simulation results of the more complex and realistic scenario we used to derive the performance evaluation in Section 3. We ran simulations for increasing total offered load to the bottleneck link, and then measured the RTT and q, p values which are later used by the model to predict the Completion Time of different flows. In particular, a couple sample values of the t_{ft} threshold were selected, and results are plotted for the 1-, 11-, 91-packet-long flows. Also in this case the model exhibits a very good prediction of the Completion Times, and therefore can be used to predict the performance experienced by users with different configuration of the FM marker.

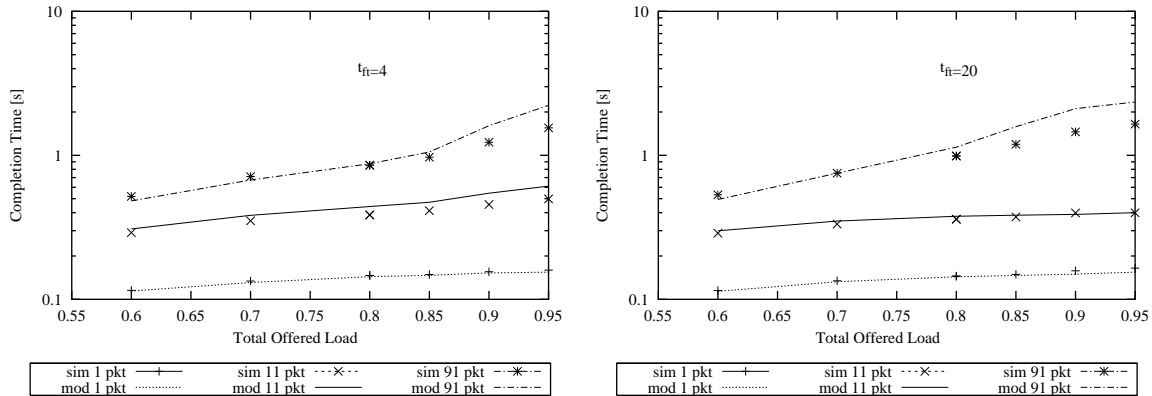


Figure 6. Comparison between the proposed model and the simulation measurements for different t_{ft}

6. Conclusions

In this paper we have discussed a lightweight, limited-state solution to the problem of fair DiffServ protection of both short- and long-lived TCP flows. In terms of performance, our solution was shown to be as effective as per-flow marking, and to greatly improve with respect to the performance of standard stateless DiffServ markers. We also proposed a simple analytical model which can accurately predict the completion time experienced by short lived flows.

REFERENCES

1. D. Black, S. Blake et al., *An Architecture for Differentiated Services*, RFC 2475, Dec. 1998.
2. S. Sahu, D. Towsley, J. Kurose, *Quantitative Study of Differentiated Services for the Internet*. IEEE Globecom'99, Rio de Janiero, pp. 1808-1817, Dec. 1999.
3. M. El-Gendy, Kang G, *Equation-Based Packet Marking for Assured Forwarding Services*, IEEE Infocom 2002, New York, NY, 23-27 Jun. 2002.
4. D. Rossi, C. Casetti, M. Mellia, *A Simulation Study of Web Traffic over DiffServ Networks*, to appear in Globecom 2002, Taipei, TW, Nov. 2002.
5. M. Mellia, I. Stoica, H. Zhang, *Packet Marking for Web traffic in Networks with RIO Routers*, Globecom 2001, San Antonio, Texas, Nov. 2001.
6. G.L. Monoco, 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, Oct. 2001.
7. W. Fang, N. Seddigh, B. Nandy, *A Time Sliding Window Three Color Marker*, RFC 2859, Jun. 2000.
8. M. Alzati, M. Bottigliengo, *Study and Simulation of Fair Markers in DiffServ Networks*, Master Thesis, Politecnico di Torino, Italy, July 2002. Available online at <http://www.tlc-networks.polito.it/mellia/papers/alzati.ps.gz>
9. Y. Chait, C.V. Hollot, V. Misra, D. Towsley, H. Zhang and J. Lui, *Providing Throughput Differentiation for TCP Flows Using Adaptive Two Color Marking and Multi-Level AQM*, IEEE Infocom 2002, New York, NY, 23-27 Jun. 2002.
10. S. Floyd, *RED: Discussions of Setting Parameters*, <http://www.icir.org/floyd/REDparameters.txt>
11. M. Mellia, I. Stoica, H. Zhang, *"TCP Model for Short Lived Flows"*, IEEE Communications Letters, vol. 6, no. 2, pp. 85-87, February 2002.
12. N. Cardwell, S. Savage, T. Anderson, *"Modeling TCP Latency"*, IEEE Infocom 2000, pp. 367-375, Tel Aviv, Israel, March 2000.
13. M.Alzati, M.Bottigliengo, C.Casetti, M.Mellia, *Modeling TCP Performance with FM3 and TWS2CM*, Technical Report, available from the authors.