On Creating Proportional Loss-Rate Differentiation: Predictability and Performance

Ulf Bodin, Andreas Jonsson, Olov Schelén

Abstract—

Recent extensions to the Internet architecture allow assignment of different levels of drop precedence to IP packets. This paper examines differentiation predictability and implementation complexity in creation of proportional loss-rate (PLR) differentiation between drop precedence levels. PLR differentiation means that fixed loss-rate ratios between different traffic aggregates are provided independent of traffic loads. To provide such differentiation, running estimates of loss-rates can be used as feedback to keep loss-rate ratios fixed at varying traffic loads. In this paper, we define a loss-rate estimator based on average drop distances (ADDs). The ADD estimator is compared with an estimator that uses a loss history table (LHT) to calculate loss-rates. We show, through simulations, that the ADD estimator gives more predictable PLR differentiation than the LHT estimator. In addition, we show that a PLR dropper using the ADD estimator can be implemented efficiently.

I. Introduction

Today's Internet supports a wide spectrum of applications with different demands on forwarding quality. The Internet community has recognized that one service only (i.e., best-effort) may not be enough to meet these demands. The Internet Engineering Task Force (IETF) is therefore designing architectural extensions to support service differentiation on the Internet. The Differentiated Services (DiffServ) architecture [2][9] includes router mechanisms for differentiated forwarding.

With DiffServ, levels of drop precedence can be assigned to IP packets. Differentiation between drop precedence levels is part of the Assured Forwarding (AF) per-hop behavior (PHB) group [8]. AF can be used to offer differentiation among rate adaptive applications that respond to packet loss (e.g., applications using TCP). The traffic of each user is tagged as being *in* or *out* of their service profiles. Packets tagged as *in*-profile are assigned lower drop precedence than those tagged as *out*-of-profile. In addition, a packet within a user's profile may be tagged with one out of several levels of drop precedence. For now, there are three levels of drop precedence defined for AF.

For AF, it is required that the levels of drop precedence are ordered so that for levels $x < y < z$, $\mathrm{P_{drop}}(x) \leq \mathrm{P_{drop}}(y) \leq \mathrm{P_{drop}}(z)$ and $\mathrm{P_{drop}}(x) < \mathrm{P_{drop}}(z)$[1]. To further refine the differentiation, it can be defined in quantitative terms. For example, the loss-rate ratios (i.e., $\mathrm{P_{drop}}(x)/\mathrm{P_{drop}}(y)$) can in case of congestion be set to a target value ≤ 1.

This paper examines proportional loss-rate (PLR) differentiation [5] in terms of predictability and implementation complexity. We grade the predictability of a PLR differentiation by studying short-term variations in loss-rate ratios between drop precedence levels at changing traffic loads and load distributions. In addition, we study if long-term loss-rate ratios achieve target loss-rate ratios at changing traffic loads. We consider a PLR differentiation *robust* if short-term loss-rate ratios have negligible variations and *capable* if long-term loss-rate ratios approximate target loss-rate ratios reasonable well at changing traffic load conditions.

[1] $\mathrm{P_{drop}}(x)$ is the drop probability for traffic at drop precedence level x.

A PLR differentiator can be divided into two modules. First, a *drop controller* decides when a packet needs to be dropped. A drop controller can, for example, perform early congestion signaling with Random Early Detection (RED) [7], or just drop packets when no buffer space is available for queuing, etc. When the drop controller has decided that a packet needs to be dropped, a *PLR dropper* selects a drop precedence level from which a packet will be dropped (to maintain the PLR differentiation), selects a packet at the victim level and removes it from the queue.

Changing traffic loads and load distributions between drop precedence levels can cause loss-rate ratios to deviate from the target loss-rate ratios. To create PLR differentiation under such conditions, a PLR dropper can use running estimates of loss-rates as feedback to adjust towards the target loss-rate ratios. For example, when the drop controller triggers a drop, a packet at the drop precedence level with the minimum normalized loss-rate (NLR) can be selected and dropped from the queue. The NLR is in [5] defined as: $\bar{l}_i/\sigma_i$ where $\bar{l}_i$ is the loss-rate and σ_i is the differentiation constant in class i. In this paper, we adopt this method of selecting the victim level when dropping packets. Using the NLR selector, we study how properties of the loss-rate estimator influence PLR differentiation predictability.

The proportional loss-rate model is proposed and motivated in [4] and [5]. In [5], a loss-rate estimator is proposed, which estimates the loss-rate by counting the number of losses at each class during a time window of M packet arrivals. One implementation of this uses a cyclic queue named a Loss History Table (LHT). The problem with this is that an appropriate value of M has to be chosen. Firstly, it has to be at least one dropped packet at all drop precedence levels in the last M packets arrived. Otherwise, the measured loss-rate will be zero for levels at which no packet is dropped in the last M arrivals. This leads to inaccurate loss-rate estimation which in turn leads to loss-rate relations which differs from the configured ones. Unfortunately, a large M makes the PLR dropper less adaptive to changing traffic loads. Hence, there is a trade-off in chosing an appropriate value of M. Large M gives capable but not robust PLR differentiation, while small M gives robust but not capable PLR differentiation.

In this paper, we define an estimator that uses average drop distances (ADDs) as estimates of loss-rates. For each drop precedence level, an ADD covers a history which length is defined in number of drops. This makes the estimator adapt the history length at each level to changing load distributions. The history covered by the ADD estimator can be set short without risking estimated loss-rates to be zero for some traffic loads. With the ADD estimator, the history length simply determines how fast changed traffic load conditions are detected. Hence, the ADD estimator does not have the same trade-off in choosing history length as the LHT estimator.

We evaluate through simulations the PLR differentiation predictability of the ADD and the LHT estimator for two levels of drop precedence. These simulations show that the trade-off in choosing M disables the LHT estimator from providing both robust and capable PLR differentiation with one single M. For the ADD estimator, weights can be found that gives both robust and capable PLR differentiation.

When designing forwarding mechanisms for Internet routers, their performance is important. Computation and/or memory intensive mechanisms make routers more expensive, which can make deployment in routers handling high bit-rates unfeasible. To examine the performance of the ADD estimator together with the NLR selector, we have implemented these mechanisms in the kernel of FreeBSD. With this implementation, only 131 clock cycles in average are needed to update three ADDs on an Intel Pentium II 350Mhz. Selecting from which of drop precedence level to drop needs 59 clock cycles in average for three levels.

The rest of the paper is structured as follows. In Sect. II, we define the ADD estimator, discuss its properties and compare these with properties of the LHT estimator. Next, in Sect. III, the need for differentiation predictability at both long and short time-scales is discussed. In Sect. IV, simulations comparing the LHT estimator and the ADD estimator are presented. In Sect. V, effective implementations of the ADD estimator and the NLR selector are described. Moreover, performance measurements for these mechanisms are presented in this section. Finally, in Sect. VI, we conclude our work.

II. Estimating Loss-Rates

In this section, we define a loss-rate estimator that uses average drop distances (ADDs). The basic properties of this estimator is discussed and compared with the properties of the loss history table (LHT) estimator [5].

TABLE I
SYMBOLS USED IN THIS PAPER

l	Aggregate loss-rate.
L_i	Drop precedence level i.
λ_i	Arrival rate at L_i.
d_i	Drop distance counter at L_i.
$d_{i_{old}}$	Old drop distance counter at L_i.
$\overline{l}_i$	Estimated loss-rate at L_i.
$\overline{d}_i$	Estimated average drop distance at L_i
$\overline{d}_{i_{old}}$	Old estimated average drop distance at L_i
σ_i	Differentiation constant for class i.
$B(t)$	Set of backlogged L_i at time t.
g_i	EWMA filter constant for L_i.
w_i	EWMA filter weigth for L_i.

A. The Average Drop Distance (ADD) Estimator

An ADD estimator calculates an average drop distance for each drop precedence level. The drop distance is the number of successfully transferred packets between two lost packets. We denote the estimated ADD at L_i as $\overline{d}_i$ and the estimated loss-rate at L_i as $\overline{l}_i = 1/\overline{d}_i$. We denote the estimated loss-rate ratio between L_i and L_j as $\overline{l}_i/\overline{l}_j$ and the target loss-rate ratio between L_i and L_j as σ_i/σ_j. The definition of normalized loss-rate (NLR) and the method for selecting precedence level when dropping a packet are both adopted from [5].

ADD estimation can be performed by computing the average drop distance over a certain number of packet drops. It is however hard to pick an appropriate number of drops to consider, especially when arrival rates and drop rates vary frequently. For quick response to changing conditions a small number of drops should be considered and for stability a large number of drops. For this reason, we instead use exponential weighted moving averages (EWMAs) to give higher weigth to recent drop distances. Although EWMA suits the purpose and can be implemented efficiently, we do not claim that it is the optimal averaging function. With EWMAs (1), the ADD estimator covers a configurable history length, $g_i, 0 \leq g_i \leq 1$, coupled to the number of rescent drops for each drop precedence level, L_i. We limit g_i to integer powers of two for efficient implementation through shift operations: $g_i = 2^{-w_i}$, where the weight w_i is a positive integer.

$$\overline{d}_{i,n} = \overline{d}_{i,n-1} \cdot (1 - g_i) + d_i \cdot g_i \tag{1}$$

Larger w_i results in more stable (i.e., more capable) estimations of $\overline{d}_i$ and longer detection times for changed trafficload conditions (i.e., less robust estimations). However, our experiments indicate that estimations of $\overline{d}_i$ is stable enough even with small values of w_i. In Sect. IV, we show that robust and capable PLR differentiation can be obtained with very small values of w_i.

The detection time is determined by both w_i and the number of drops per time unit. For robustness, changed traffic load conditions should be detected equally fast at all drop precedence levels. If $\overline{d}_i$ is only updated upon packet drops at L_i, different arrival rates and loss-rates between drop precedence levels causes different update fequency and detection times between these levels. For example, assume that the actual loss-rate is higher at L_k than at L_j. Then, $\overline{d}_k$ is updated more often than $\overline{d}_j$ and changed traffic load conditions is thus detected faster at L_k than at L_j. Moreover, if the load suddenly decreases and stays at the lower level, $\overline{d}_j$ may not be updated at all. This is because $\overline{l}_j/\sigma_j$ becomes larger than $\overline{l}_k/\sigma_k$, which causes drops to be strictly given to traffic at L_k. We refer to this problem as *update locking*.

To avoid the risk for update locking and to make detection times similar between drop precedence levels, we also recalculate $\overline{d}_j$ at drop precedence levels, j, which was not targeted for a drop. We do this by restoring $\overline{d}_j$ to the value it had before at the time of the previous drop at L_j and then recalculate $\overline{d}_j$ with all new arrivals at L_j added to the last drop distance at L_j. Not only do we get a more up to date estimate of $\overline{d}_j$, but also we solve the update locking problem since $\overline{d}_j$ goes towards infinity with $d_j + d_{j_{old}}$ (see Fig. 1).

Equal weights, w_i, for all drop precedence levels makes the detection time at L_i shorter (i.e., $\frac{\sigma_i \cdot \lambda_i}{\sigma_j \cdot \lambda_j}$ times shorter) than at L_j.

```
Packet arrival at $L_k$:
    $d_k$++
Packet drop at $L_k$:
    $\overline{d}_{k_{old}} \leftarrow \overline{d}_k$
    $\overline{d}_k \leftarrow \overline{d}_k \cdot (1 - 2^{-w_k}) + d_k \cdot 2^{-w_k}$
    $d_{k_{old}} \leftarrow d_k$
    $\forall i \in B(t) \setminus \{k\}$:
        $\overline{d}_i \leftarrow \overline{d}_{i_{old}}$
        $d_{i_{old}} \leftarrow d_{i_{old}} + d_i$
        $\overline{d}_i \leftarrow \overline{d}_i \cdot (1 - 2^{-w_k}) + d_{i_{old}} \cdot 2^{-w_k}$
    $\forall i \in B(t)$:
        $d_i \leftarrow 0$
Selecting $L_k$ for next drop:
    $k \leftarrow \arg\max_{i \in B(t)} \overline{d}_i \cdot \sigma_i$
```

Fig. 1. The algorithm for the ADD estimator and the NLR selector

To compensate for this, separate weights for each drop precedence level can be applied such that (2) is met as closely as possible[2]. Equation (2) is based on the closed form expression for EWMA (1).

$$(1 - 2^{-w_i}) \quad = \quad (1 - 2^{-w_j})^{\frac{\sigma_j \cdot \lambda_j}{\sigma_i \cdot \lambda_i}} \qquad (2)$$

The algorithm for the ADD estimator and the NLR selector is shown in Fig. 1. Note that the inverse NLR, $\overline{d}_i \cdot \sigma_i$, instead of the NLR, $\overline{l}_i / \sigma_i$, is used to select from which L_i to drop. Hence, at congestion we drop a packet at the drop precedence level with the *maximal* inverse NLR instead of the *minimal* NLR as done in [5].

With the algorithm shown in Fig. 1, $\overline{d}_i$ does not change if no packets arrive at L_i and will therefore be invalid after a idle period at this drop precedence level. This becomes a problem if $\overline{d}_i \cdot \sigma_i$ is larger than the inverse NLR for other levels. The first packets arriving at L_i immediately after the idle period will then be dropped until $\overline{d}_i \cdot \sigma_i$ becomes smaller than the inverse NLR for some other level. Similarly, the first packets arriving at L_i immediately after an idle period will not be dropped if $\overline{d}_i \cdot \sigma_i$ is smaller than the inverse NLR for some other level. Hence, loss-rate ratios can temporarily be larger or less than the target loss-rate ratios. We refer to this problem as *invalid ADDs*.

Dropping the first packets arriving after an idle period can be devastating since TCP sources perform an exponential back-off when loosing SYN packets. Due to the exponential back-off, it can take considerable time for $\overline{d}_i$ to decrease since no packets arrive at L_i. We solve the invalid ADDs problem by updating $\overline{d}_i$ to a value calculated from known ADDs at other drop precedence levels. The update is made if no packet has arrived after *maxidle* updates of ADDs for other levels (Fig. 2).

The method for detecting and updating idle drop precedence levels shown in Fig. 2 can cause deviations of loss-rate ratios from target loss-rate ratios. For example, say that L_i is frequently idle for periods long enough to trigger an update and that each update decreases $\overline{d}_i \cdot \sigma_i$. Moreover, say that the active periods are short and that $\overline{d}_i \cdot \sigma_i$ therefore does not reach

[2]Equation (2) cannot be met exactly since w_i is a positive integer.

```
Packet arrival at $L_k$:
    $idle_k \leftarrow 0$
Packet loss at $L_k$:
    $\forall i \in B(t) \setminus \{k\}$:
        if $d_i = 0$ and $idle_i$++ $>$ maxidle:
            $a \leftarrow \arg\min_{j \in B(t)} \overline{d}_j \cdot \sigma_j$
            $\overline{d}_i \leftarrow \overline{d}_a \cdot \frac{\sigma_a}{\sigma_i}$
```

Fig. 2. Method for detecting and updating idle drop precedence levels

the actual inverse NLR before L_i gets idle again (i.e., although very few packets are dropped at L_i, $\overline{d}_i$ does not increase enough to reflect the loss-rate at L_i). This may cause the loss-rate ratio between L_i and the next lower level to be less than the target loss-rate ratio between these levels. We refer to this problem as *frequent updates*. To avoid the frequent updates problem, *maxidle* should be set large. We recommend setting *maxidle* to trigger updates after idle periods of several seconds. This mechanism is disabled in the simulations presented in Sect. IV.

Distributions in arrival rates between drop precedence levels (i.e., λ_j / λ_i) at congested links is usually unknown and may change rapidly for bursty traffic patterns. However, different arrival rates is a severe problem only if the loss-rate changes rapidly with changing traffic loads. This is often the case for pure FIFO queues, but not for Random Early Congestion (RED) [7] managed queues. RED smoothes the loss-rate using a low pass filter (e.g., EWMA). We take advantage of the smooth changes in loss-rates provided by RED and do not compensate for different arrival rates (i.e., we set $\lambda_j / \lambda_i = 1$). When loss-rates are controlled with RED, differences in detection time caused by different arriving rates have limited effect on the PLR differentiation offered. This is shown in Sect. IV where simulations are presented.

A consequence of using RED to smooth loss-rates is that the ADD estimator depends on proper operation of RED. Recent studies of RED have shown that the average queue length and thus the loss-rate can oscillate under certain conditions (the discontinuity in the standard RED drop function[3] and/or some combinations of link bandwidth, average packet size and load levels can cause such oscillations [11]). Based on these studies, a new active queue management (AQM) mechanism is developed that gives more robust loss-rates than RED [3]. The ADD estimator should gain from the smoother loss-rates provided by this new AQM mechanism. However, in this paper we evaluate the ADD estimator with RED without the gentle modification.

B. The Loss History Table (LHT) Estimator

The loss history table (LHT) estimator is defined in [5]. The estimated loss-rate $\overline{l}_i$ is the number of drops at L_i in the last M arrivals divided by M. The cyclic queue used to count drops is named loss history table (LHT). M has to be large enough to always cover at least one drop at all drop precedence levels. Otherwise, acceptable estimation accuracy is not obtained since $\overline{l}_i$ occasionally becomes zero. Equation (3) gives a lower bound

[3]The standard drop function in RED jump from the maximal drop probability (e.g., 0.1) to 1 instantly. This discontinuity is however removed with the gentle modification of RED [6].

on M [5]. N is the number of drop precedence levels supported and $m = \arg\min_{0 \le i \le N-1} \lambda_i \cdot \sigma_i$.

$$M_{min} = \frac{\sum_{i=0}^{N-1} \frac{\lambda_i \cdot \sigma_i}{\lambda_m \cdot \sigma_m}}{l} \quad (3)$$

M should be larger than the lower bound given by (3) in order to to provide capable PLR differentiation. This is shown through simulations in [5]. For instance, bursty traffic can give considerable variations in the arrival rate and the loss-rate over short time-scales, which will degrade the differentiation if M too small. Unfortunately, large M makes the detection of changed traffic load conditions slow. Hence, there is a trade-off in selecting M. Larger M gives more capable, but less robust PLR differentiation and smaller M less capable, but more robust PLR differentiation.

C. Comparison

The ADD estimator provides both robust and capable PLR differentiation with one configuration. In contrast to the LHT estimator, it provides accurate loss-rate estimation by always encountering several drops at every drop precedence level. Hence, without risking inaccurate loss-rate estimation with incapable PLR differentiation as result, the ADD estimator can be configured to encounter few drops to detect changed traffic load conditions rapidly. The LHT estimator cannot be configured to provide both accurate loss-rate estimation and rapid detection of changed traffic load conditions. Fast detection of changed traffic load conditions is needed to provide robust PLR differentiation.

If a small number of drops is covered by the loss history, the loss-rate estimation becomes unstable at short time-scales. Such unstable loss-rate estimation can cause variations in actual loss-rate ratios at short time-scales and deviations of actual loss-rate ratios from target loss-rate ratios at long time-scales. The EWMA makes it hard to give the parameters w_i a clear physical interpretation, as opposed to the LHT estimator, where M corresponds to the number of packet arrivals. However, this is not necessary to configure the ADD estimator appropriately. We show in Sect. IV that by using $w_0 = 1$ and $w_1 = 4$[4], the ADD estimator provides robust and capable PLR differentiation between L_0 and L_1 for $\sigma_0 = 1$ and $\sigma_1 = 10$.

III. Measuring Loss-Rates

In this section, we discuss over which time-scales loss-rate ratios are likely to be measured by network operators and to be perceived by users. Network operators may monitor loss-rates by polling routers periodically using SNMP or command line interfaces. The overhead associated with periodic polling makes it appropriate to monitor loss-rates over time-scales in order of minutes rather than seconds. However, users are likely to perceive loss-rate ratios over time-scales spanning from few seconds to several minutes.

PLR differentiation allows individual users to choose a service that provides an appealing balance between forwarding quality and cost. With PLR differentiation, a user can dynamically switch between levels of drop precedence to find a level with a loss-rate low enough for the application used. A user can begin tagging all the packets with a high drop precedence level. If the loss-rate at this level is considered unacceptably high after a period, the user can switch to a lower drop precedence level. Eventually, the user should find a level that provides a loss-rate adequate for the user's needs. Hence, the user does not have to pay for additional and unneeded forwarding quality.

To make the result of switching from one level of drop precedence to another level predictable, the PLR differentiation needs to be robust and capable. Loss-rate ratios measured over several minutes need to closely approximate target loss-rate ratios. Otherwise, users cannot predict the result of switching drop precedence level. Moreover, loss-rate ratios measured over a few seconds need to have negligible variations. This is to make the result of switching drop precedence level immediately notable to users.

IV. Simulations

In this section, we present simulations evaluating the predictability of the PLR differentiation created with the LHT estimator and the ADD estimator respectively. The simulations are made with the network simulator (ns) [10]. Sect. IV-A describes the simulation setup. We study PLR differentiation at a time-scale of two minutes in Sect. IV-B and at a time-scale of five seconds in Sect. IV-C.

A. Simulation Setup

For the simulations, a topology with ten hosts (s0, ..., s9), ten receivers (r0, ..., r9) and two routers (A and B) is used. The routers are connected via a 50 Mbps link with 20 ms delay (Fig. IV-A). A PLR dropper supporting two levels of drop precedence is used to differentiate traffic at this link. The target loss-rate ratio σ_1/σ_0 is set to 10 times (i.e., the loss-rate at drop precedence level 1 is targeted to be 10 times higher than the loss-rate at level 0). The drop controller is RED and the drop strategy is Drop-Tail[5]. The configuration of RED is: min threshold 70 packets, max threshold 210 packets and max drop probability 10 percent.

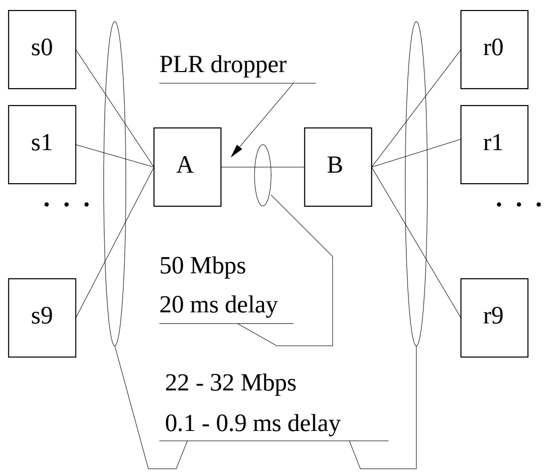
The bit-rates of links connecting hosts and receivers to routers are reconfigured with uniformly distributed random values between 22 and 32 Mbps once every two simulated seconds. The delays of these links are reconfigured equal often with uniformly distributed random values between 0.1 and 0.9 ms. Similar values are used in [1] to emulate switched Ethernet. A positive consequence of making these reconfigurations is that synchronization affects among TCP connections get reduced[6].

Each host (s0, ..., s9) has three TCP Reno connections with each receiver (r0, ..., r9) (i.e., 300 connections are established over link A-B). The receivers use delayed ACKs. MTU is 1460 bytes. The TCP connections are established randomly within the first 10 simulated seconds. These random variables are uniformly distributed. Using these connections, the receivers download data from Pareto distributed ON-OFF sources at the

[4]With $w_0 = 1$ and $w_1 = 4$ the equality in (2) is approximately satisfied when arrival rates are equal for L_0 and L_1.

[5]Packets are removed from the end of the queue for the drop precedence level.

[6]The random drops made by RED also reduce the risk of having TCP flows synchronize.



captionSimulated topology.

hosts. The scale parameter for the Pareto distribution is 1.5, the average length of *ON* periods is set to 50 ms and the average length of *OFF* periods is set to 950 ms. The rate of *ON* periods for each source is set to 490 kbps. This generates a highly variable traffic load causing loss-rates in between 1.72 and 5.15 percent at link A-B when measured over two minutes (the simulations presented in Sect. IV-B). When measured over five seconds (the simulations presented in Sect. IV-C), loss-rates are in between 1.15 and 5.78 percent at this link.

For all simulations, a warm-up period of 60 simulated seconds is used to let the congested queue at router A and the loss-rate estimator examined stabilize. After these 60 seconds, counters for the number of packet arrivals and drops at each of the two levels of drop precedence are initialized to measure loss-rate ratios. In Sect. IV-B and IV-C, loss-rate ratios are plotted with a log 10 scale at the y-axis. The log scale is chosen to view deviations of loss-rate ratios equally independent on whether they are larger or less than the target loss-rate ratio.

B. Long-Term PLR Differentiation

For each of the loss-rate estimators, we examine their long-term PLR differentiation predictability with 19 simulations. The distribution in number of TCP connections at the two levels of drop precedence is changed between simulations. At drop precedence level 0 (L_0), the number of TCP connections is varied between 15 and 285 in steps of 15. At L_1, the number of TCP connections is varied between 285 and 15 in steps of 15. The x-axis is graded with the fraction of all packet arrivals at L_0. Each simulation is 120 seconds long.

Figure 3 shows simulation results when using the LHT estimator. For these simulations, (3) gives $M_{min} \approx 5000$ packets when 15 TCP connections are at L_0[7]. With this distribution, about 5 percent of all packet arrivals are at L_0. As discusseded in Sect. II-B, M should however be set larger than M_{min}. We present simulations with $M = 5000$, $M = 10000$ and $M = 25000$ packets. At a bit-rate of 50 Mbps, 10000 packets of size 1460 bytes are forwarded in 2.336 seconds. Hence, with $M = 10000$ packets, the LHT estimator can be expected to adapt to changing traffic load conditions faster than in five seconds. With $M = 25000$ packets, this adaptation should be slower than in five seconds (25000 packets of size 1460 bytes are forwarded in 5.84 seconds).

[7] $M_{min} = 3767$ packets for $M = 5000$, $M_{min} = 5997$ packets for $M = 10000$ and $M_{min} = 5059$ packets for $M = 25000$.

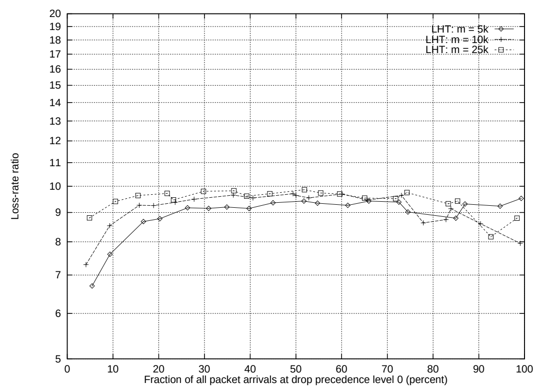


Fig. 3. Actual loss-rate ratios measured over two minutes (the LHT estimator).

In Fig. 3, it can be seen that at low arrival fractions at L_0, loss-rate ratios is less than 9 for all M simulated. As expected, higher M gives higher loss-rate ratios at such low fractions. With too few packets at L_0, the LHT occasionally falsely estimates $\bar{l}_i$ to zero. The dropper will then select L_0 for a packet drop. At high arrival fractions at L_0, loss-rate ratios varies and go below 9 for all M simulated. When no packets arrive at L_1, packets can only be dropped at L_0 since the queue at L_1 is empty. Consequently, when a packet should be dropped at L_1 to increase the loss-rate ratio, it will have to be dropped at L_0 instead. With close to 100 percent of all packet arrivals at L_0, packet drops are forced to L_0 because of an empty queue at L_1 in 4.4 percent of all drops for $M = 5000$ packets, 8.3 percent of all drops for $M = 10000$, and 22 percent of all drops for $M = 25000$.

A forced packet drop at L_0 decreases the loss-rate ratio. It takes relatively long time to repair this since a drop at L_0 has a larger impact than a drop at L_1. If a forced drop is not repaired before M arrivals, the loss-rate ratio will be permanently too low. As can be observed in Fig. 3, larger fractions of forced drops at L_0 decreases the loss-rate ratio.

Figures 4 and 5 show simulation results when using the ADD estimator. We present simulations with (w_0, w_1) = (1,2), (1,3), (1,4), (2,3), (2,4) and (2,5). The three first configurations are shown in Fig. 4 and the last three configurations in Fig. 5. (w_0, w_1) = (1,4) and (2,5) approximates the equality in (2) when arrival rates are equal at L_0 and at L_1.

Loss-rate ratios are degraded for low arrival fractions at L_0 with the ADD estimator (Figs. 4 and 5). This is because the ADD estimator detects an increasing loss-rate more rapidly for L_1 when there are more arrivals at L_1 than at L_0. This property of the ADD estimator is discussed in Sect. II-A. Without RED smoothing actual loss-rates, the problem of different detection times cause severe degradations in loss-rate ratios.

For configurations not satisfying (2) (i.e., (w_0, w_1) = (1,2), (1,3), (2,3) and (2,4)), loss-rates are lower than for configurations that do (Figs. 4 and 5). Lower w_1 than given by (2) implies that changes in loss-rates are detected faster at L_1 than at L_0 except for high arrival fractions at L_0. For example, with (w_0, w_1) = (1,2), λ_0/λ_1 needs to be 4.15 to satisfy (2). This means that

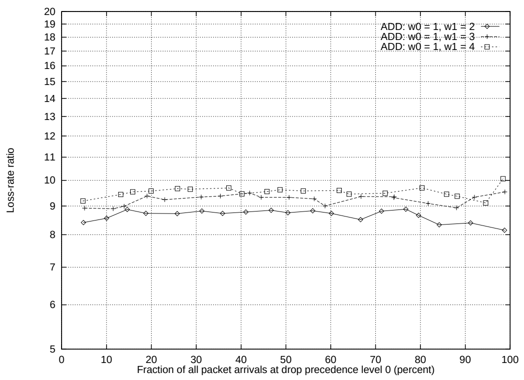


Fig. 4. Loss-rate ratios measured over two minutes (the ADD estimator, configuration set 1).

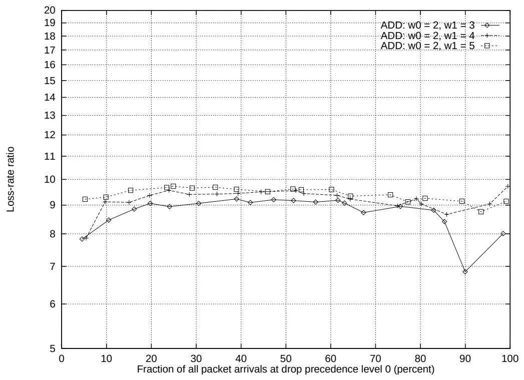


Fig. 5. Loss-rate ratios measured over two minutes (the ADD estimator, configuration set 2).

loss-rates are detected faster at L_1 than at L_0 for all arrival fractions at L_0 up to 80 percent. This percentage is similar with (w_0, w_1) = (2,3) as with (w_0, w_1) = (1,2). With (w_0, w_1) = (1,3) and (2,4), it is about 65 percent.

For high arrival fractions at L_0, the fraction of forced drops from L_0 gets high for all configurations of the ADD estimator (i.e., (w_0, w_1) = (1,2) gives up to 6.4 percent forced drops, (1,3): 13 percent, (1,4): 21 percent, (2,3): 12 percent, (2,4): 24 percent and (2,5): 32 percent). This suggests that loss-rate ratios should be degraded for high arrival fractions at L_0. However, since the invalid ADDs problem cause increases in loss-rate ratios (Sect. II-A), the degradation expected from the high fractions of forced drops from L_0 get balanced out so that loss-rate ratios approximates the target loss-rate ratio of 10 (Figs. 4 and 5). This explains the increase in loss-rate ratios with larger w_1 in these simulations.

In Figs. 4 and 5, it can be seen that the ADD estimator provides capable PLR differentiation for low arrival fractions at L_0 with the configurations that satisfy (2) (i.e., for (w_0, w_1) = (1,4) and (2,5)). The LHT estimator needs large M to provide capable PLR differentiation for arrival fractions less than 15 percent (i.e., $M = 25000$ packets).

The (1,4) configuration of the ADD estimator is preferable before the (2,5) configuration since it gives fewer forced drops. Moreover, the invalid ADDs problem is less severe with small weights. Fewer forced drops and a less severe invalid ADDs problem should make the PLR differentiation more robust.

C. Short-Term PLR Differentiation

For each of the two loss-rate estimators evaluated, we examine their short-term PLR differentiation predictability. The simulations run for 360 seconds after the warm-up period. Loss-rate ratios are measured 5 seconds interval. At 180 and 300 seconds, the distribution in number of TCP connections between L_0 and L_1 is changed. In the first 60 seconds after the warm-up, there are 15 TCP connections at L_0 and 285 TCP connections at L_1. In the next 120 seconds of the simulations, there are 150 TCP connections at each drop precedence level. Finally, in the last 120 seconds of the simulations, there are 285 TCP connections at L_0 and 15 TCP connections at L_1.

For this scenario, we have used the same parameters for the LHT estimator and the ADD estimator as for the simulations presented in Sect. IV-B (i.e., M = 5000, 10000, 25000 and (w_0,w_1) = (1,2), (1,3), (1,4), (2,3), (2,4) and (2,5)). Fig. 6 shows simulation results for the LHT estimator with M = 5000 and 10000 packets and Fig. 7 for the LHT estimator with M = 10000 and 25000 packets. Thereafter, Fig. 8 through Fig. 11 show simulation results for the different configurations of the ADD estimator.

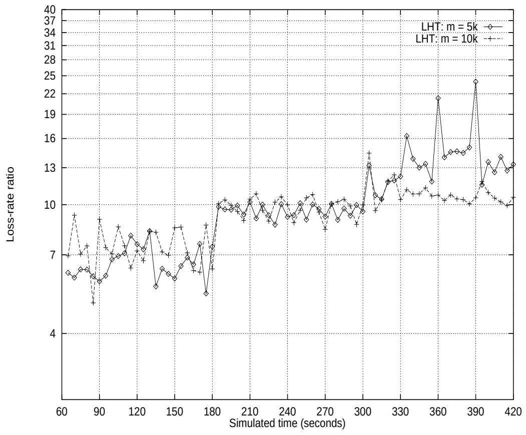


Fig. 6. Loss-rate ratios measured over five seconds (M = 5000 and 10000).

When the arrival fraction at L_0 is low or high, loss-rate ratios are closer to the target loss-rate ratio with larger M (Figs. 3, 6 and 7). Using small M, chances are that some levels have not ben targeted for a drop in the last M arrivals, causing the estimated loss-rate to be zero for levels with low arrival rate. With

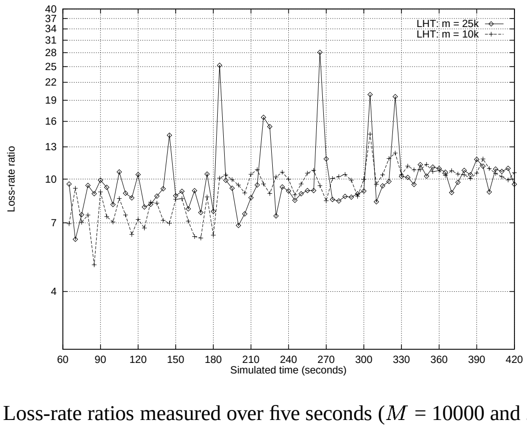


Fig. 7. Loss-rate ratios measured over five seconds (M = 10000 and 25000).

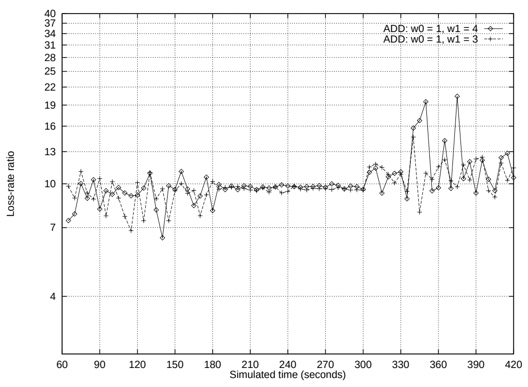


Fig. 8. Loss-rate ratios measured over five seconds ((w_0,w_1) = (1,4) and (1,3)).

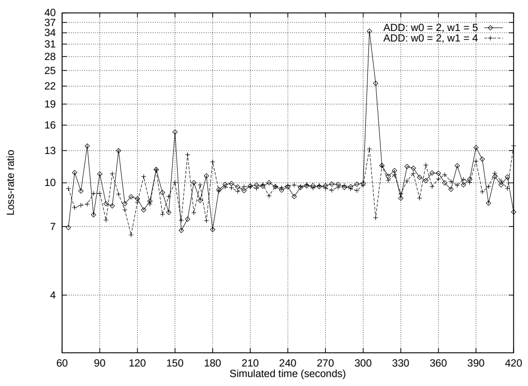


Fig. 9. Loss-rate ratios measured over five seconds ((w_0,w_1) = (2,5) and (2,4)).

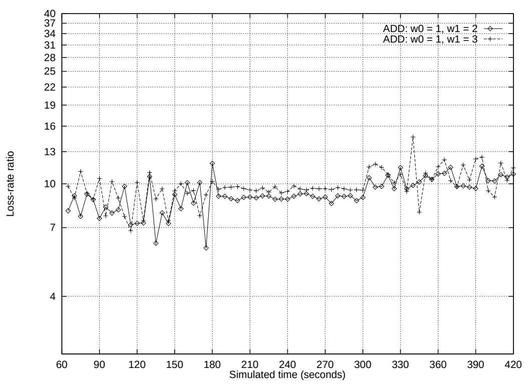


Fig. 10. Loss-rate ratios measured over five seconds ((w_0,w_1) = (1,2) and (1,3)).

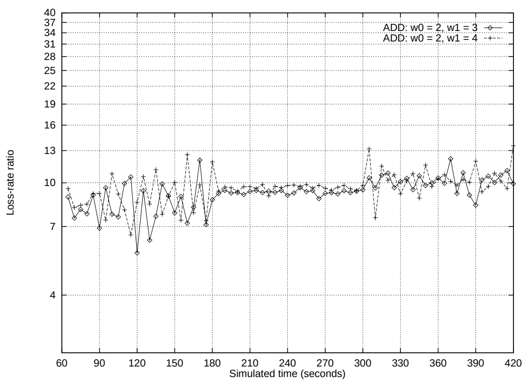


Fig. 11. Loss-rate ratios measured over five seconds ((w_0,w_1) = (2,3) and (2,4)).

$M = 5000$ packets, this happens frequently for both low and high arrival fractions at L_0, but only for low arrival fractions at L_0 with $M = 10000$ packets. With $M = 25000$ packets, estimated loss-rates become seldom zero for levels with low arrival rate and loss-rate ratios therefore better approximate the target loss-rate ratio. Figures 6 and 7 also shows that large M causes large variation in loss-rate ratios. This is because larger M gives slower detection of changed traffic load conditions.

For low arrival fractions at L_0, the variation in loss-rate ratios is larger for higher weights (before 180 seconds in Figs. 8 and 9). Nevertheless, this variation is similar for the ADD estimator with (w_0,w_1) = (1,4) and for the LHT estimator with $M = 5000$ packets (Fig. 6). The variation in loss-rate ratios is smaller with $M = 5000$ packets than with $M = 10000$ or 25000 packets.

For high arrival fractions at L_0 (after 300 seconds in Figs. 8 and 9), the variation in loss-rate ratios is higher with the ADD estimator than with the LHT estimator if a configuration satisfying (2) is used. The high variation with the ADD estimator is caused by the invalid ADDs problem described in Sect. II-A. Although all arriving packets at L_1 are dropped, the loss-rate is increased slowly due to very few packet arrivals at this level.

The variation in loss-rate ratios for high arrival fractions at L_0 can be decreased by activating the method for detecting and updating idle levels shown in Fig 2. This method cannot however eliminate the variation in loss-rate ratios for high arrival fractions at L_0. Moreover, the method for detecting and updating idle levels can cause long-term loss-rate ratios to deviate from the target loss-rate ratio. This is because of the frequent updates problem described in Sect. II-A.

D. Summary of Simulation Results

In Sect. IV-B, we examine differentiation predictability at a time-scale of two minutes for the ADD estimator and the LHT estimator respectively. We show that the LHT estimator needs large M to provide capable PLR differentiation for arrival fractions less than 15 percent (i.e., $M = 25000$ packets). The ADD estimator provides capable PLR differentiation for low arrival fractions at L_0 with configurations that satisfy (2) (i.e., for (w_0, w_1) = (1,4) and (2,5)).

Next, in Sect. IV-C, we examine differentiation predictability at a time-scale of five seconds for the two loss-rate estimators evaluated. When configured for equal arrival rates at both drop precedence levels (i.e, for (w_0, w_1) = (1,4) and (2,5)), the variation in loss-rate ratios is similar or lower with the ADD estimator than with the LHT estimator (i.e., for M = 25000 packets). With M = 10000 packets, this variation is higher with the ADD estimator than with the LHT estimator. Such a configuration of the LHT estimator does not however give a capable PLR differentiation for low arrival fractions at L_0. Hence, the LHT estimator cannot provide both capable and robust PLR differentiation with one configuration. Since the ADD estimator can be both capable and robust with one configuration, we consider it more predictable than the LHT estimator.

E. Configuration Recommendations

The robustness of the PLR differentiation can be improved with the ADD estimator by setting (w_0,w_1) = (1,3) or (w_0,w_1) = (2,4). Such configurations give robust PLR differentiation at high fractions of all traffic at L_0, but less capable PLR differentiation at most traffic distributions.

Simulations with different link speeds, RTTs and number of TCP flows indicate that the above given configurations are not particularly sensitive to these parameters[8]. The ADD estimator may however be sensitive to scenarios in which the EWMA averaging function of RED gives an oscillating loss-rate (the discontinuity in the standard RED drop function and/or some combinations of link bandwidth, average packet size and load levels can cause such oscillations [11]). Our simulations do not cover such scenarios since the packet size is fixed to 1460 bytes and all TCP flows have similar RTTs. We consider the issue of analyzing oscillations of loss-rates caused by RED, evaluating different averaging functions for RED and the ADD estimator, and examining new AQM mechanisms that gives smother loss-rates as for further studies.

Based on our simulations, we recommend to set $w_0 = 1$ and w_i for other L_i using (2). If improved robustness is required at very high fractions of all traffic at low drop precedence levels and less capable PLR differentiation can be accepted, lower values of w_i or higher values of w_0 can be used than those given by (2).

V. Implementation Complexity

In this section, we describe an efficient implementation of the ADD estimator and the NLR selector. We also include an evaluation of the computational cost of an implementation on a test platform. This evaluation show that the overhead introduced by an implementation of ADD is small compared to other tasks a router need to perform. The computational cost of these mechanisms $\in O(n)$ (linear complexity), where n is the number of drop precedence levels. Since n is expected to be small (e.g., n = 3 for DiffServ AF), we do not consider this to be significant.

The ADD estimator is designed to allow implementation without floating-point arithmetics, divisions, or multiplications.

To further improve the performance of the differentiation dropper, the drop distance counter, d_i, is increased with σ_i instead of 1 upon packet drops at drop precedence level i. This gives the inverse NLR, $\overline{d}_i \cdot \sigma_i$, without multiplications (4).

$$\overline{d}_{i,n} \cdot \sigma_i = \overline{d}_{i,n-1} \cdot \sigma_i \cdot (1 - g_i) + d_i \cdot \sigma_i \cdot g_i \qquad (4)$$

The differentiation constants, σ_i, can be treated as fixed point decimal numbers so that relations with decimal precision can be configured by scaling σ_i with a factor of 10^p, where p is the desired number of decimal positions.

VI. Conclusions

In this paper we define a loss-rate estimator based on average drop distances (ADDs). The ADD estimator is designed to offer *robust* and *capable* proportional loss-rate (PLR) differentiation at varying traffic loads. We consider a PLR differentiation *robust* if short-term loss-rate ratios have negligible variations and *capable* if long-term loss-rate ratios approximate target loss-rate ratios reasonable well at changing traffic load conditions.

[8]Due to limited space, we do not show simulations with different link speeds, RTTs and number of TCP flows.

We evaluate, through simulations, the PLR differentiation predictability of the ADD estimator and an estimator implemented with a loss history table (LHT) for two levels of drop precedence. These simulations show that the LHT estimator cannot provide both robust and capable PLR differentiation with one single M. For large M, the target loss-rate ratio is well approximated by the loss-rate ratio at long time-scales. However, for such M, the short-term loss-rate ratio can vary appreciably when traffic load varies.

For small M, low variation in the short-term loss-rate ratio is obtained at varying traffic loads, but it does not reach the target loss-rate ratio at long lime-scales. For the ADD estimator, weights can be found that gives both robust and capable PLR differentiation (i.e. the short-term loss-rate ratio has low variation and the long-term loss-rate ratio approximates the target loss-rate ratio). The ADD estimator requires however that the actual loss-rate is smooth (e.g. by using RED). Without proper smoothing of the actual loss-rate, the ADD estimator may not give predictable PLR differentiation.

To evaluate the performance of the ADD estimator, we have implemented a PLR dropper using the ADD estimator in the kernel of FreeBSD. With three levels of drop precedence supported, this dropper needs only 131 clock cycles in average to update ADDs and 59 clock cycles in average to select from which precedence level to drop on an Intel Pentium II 350Mhz.

References

[1] Polly Huang Anja Feldmann, Anna C. Gilbert and Walter Willinger. Dynamics of ip traffic: A study of the role of variability and the impact of control. *ACM Computer Communications Review,* October 1999.

[2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated service. Request for Comments (Informational) 2475, Internet Engineering Task Force, December 1998.

[3] Don Towsley C.V. Hollot, Vishal Misra and Wei-Bo Gong. On designing improved controllers for aqm routers supporting tcp flows. *Proceedings of Infocom 2001,* April 2001.

[4] Constantinos Dovrolis and Parameswaran Ramanathan. A case for relative differentiated services and the proportional differentiation model. *IEEE Network,* 13(5):26–34, September/October 1999.

[5] Constantinos Dovrolis and Parameswaran Ramanathan. Proportional differentiated services, part II: Loss rate differentiation and packet dropping. In *Proceedings of IWQoS'2000,* Pittsburgh, June 2000.

[6] Sally Floyd. Recommendations on using the gentle variant of red. Notes, March 2000.

[7] Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking,* 1(4):397–413, August 1993.

[8] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. Assured forwarding PHB group. Request for Comments 2597, Internet Engineering Task Force, June 1999.

[9] K. Nichols, S. Blake, F. Baker, and D. Black. Definition of the differentiated services field (DS field) in the IPv4 and IPv6 headers. Request for Comments (Proposed Standard) 2474, Internet Engineering Task Force, December 1998.

[10] UCB/LBNL/VINT. Network simulator—ns (version 2.1b5). http://www-mash.cs.berkeley.edu/ns/, 1999.

[11] Wei-Bo Gong Vishal Misra and Don Towsley. Fluid-based analysis of a network of aqm routers supporting tcp flows with an application to red. *ACM Computer Communications Review,* October 2000.