How to Make Assured Service More Assured

Wei Lin, Rong Zheng, Jennifer C. Hou

Department of Electrical Engineering
The Ohio State University
Columbus, OH 43210
{linw, zhengr,jhou}@ee.eng.ohio-state.edu

Abstract In this paper, we study why the current assurance services (AS) architecture with profilers/markers at edge routers and with queue management mechanisms (RIO) at core routers cannot achieve throughput assurance and fairness. Based on the findings of the simulation study, we propose an enhanced version of the time sliding window (TSW) profiler, called the *enhanced TSW* (ETSW). We also design two enhanced versions of the RIO queue management mechanism, called respectively, the (r, RTT)-*adaptive* algorithm and the *dynamic RIO* (DRIO) algorithm, based on rigorous, analytical reasoning. To validate the proposed design, we implement the proposed mechanisms, along with the 2-window TCP scheme [3], the three color-marker scheme [4, ?], and the CSFQ scheme [?], in ns-2 [8], and examine their behavior under a variety of network topologies and traffic sources. The simulation results indicate that both DRIO and (r, RTT)-adaptive algorithms, when combined with ETSW, do fulfill more satisfyingly the throughput assurance and fairness requirements, especially under the case that AS flows require different target rates, incur different round trip times, or co-exist with non-responsive UDP flows.

Index Terms — Differentiated services, profilers, queue management mechanisms.

1 Introduction

Motivated by the enriching content of multimedia applications and by the changing requirements and functionality at the user sides, network services with different levels of QoS for multiple-point applications have become increasingly demanding. Recently, the IETF DiffServ (Differentiated Services) working group has proposed, based on the two proposals by Clark [?] and Jacobson [?], a differentiated services architecture that relies on packet tagging and lightweight router support to provide *premier* and *assured* services that extend beyond best effort [5].

In the DiffServ architecture, a source specifies a *service class* (e.g., *premium*, *assured*, or *best effort*) and a *service profile*, that indicates the amount of traffic that the sender negotiates to send in the specified class. In particular, the class of assured services (AS) is intended to give the customer the assurance of a minimum throughput (called the *target rate*), even during periods of congestion, while allowing it to consume, in some fair manner, the remaining bandwidth when the network load is low. More specifically, the two major requirements of assured services are:

Throughput assurance: Each flow should receive its subscribed target rate on average;

Fairness: The surplus (unsubscribed) bandwidth is evenly shared among AS and best-effort flows.

There are two major components to realize the above objectives in the AS architecture: (i) the packet marking mechanism (which includes meters and markers) at edge routers to classify packets as *in-profile* or *out-of-profile* prior to their entering the network, and (ii) the queue management mechanism used at core routers to differentiate and forward packets based on their service class and marking. In the current AS architecture, the token bucket and the time sliding window (TSW) are the two most commonly used profilers, and the RED with IN and OUT (RIO) has received the most attention among all the router mechanisms proposed for assured services. Although the current AS architecture is simple and does not require per-flow state information, as reported in [?, ?], it cannot meet the throughput assurance and fairness requirements under certain cases.

In this paper, we study why the current AS architecture cannot achieve throughput assurance and fairness. Based on the findings of the simulation study, we then propose an enhanced version of TSW, called the *enhanced TSW* (ETSW), to eliminate the drawbacks of both the token bucket and the TSW profiler. We also design two enhanced versions of the RIO queue management mechanism, called respectively, the (r, RTT)-*adaptive* algorithm and the *dynamic RIO* (DRIO) algorithm, to improve the degree of throughput assurance and fairness for assured services. Both queue management mechanisms are derived based on rigorous, analytical reasoning. To validate the proposed design, we implement the proposed mechanisms, along with the other

The work reported in this paper was supported in part by NSF under Grant No. ANI-9804993.

existing mechanisms, in ns-2 [8], and examine their behavior under a variety of network topologies and traffic sources. The simulation results indicate that both DRIO and (r, RTT)-adaptive algorithms, when combined with ETSW, do fulfill more satisfyingly the throughput assurance and fairness requirements. The simulation results also indicate the proposed schemes outperform two-window TCP [**?**], three-color marker [4, **?**], and CSFQ [7], under most of the cases.

The rest of the paper is organized as follows. After the introductory part, we study in Section 2 the problems associated with each network component – the profiler at an edge router and the queue management mechanism at a core router inside the network – in the AS architecture. In Section 3, we propose an enhanced version of TSW, ETSW. In Sections 4–5, we present two enhanced versions of the RIO queue management mechanism, namely the (r, RTT)-adaptive algorithm and the dynamic RIO (DRIO) algorithm. In Section 6, we evaluate, and compare the performance of, the proposed schemes against RIO and the other schemes. We conclude the paper with Section 7.

2 Problems Associated with the Current Assured Services Architecture

Setup of the Simulation Environment: Two types of topologies are used in our simulations. First, we consider a simple topology with TCP Reno hosts connecting to their respective destinations via one common bottleneck link shown in Fig. 1. Second, we consider a multiple-bottneck senario.In fig. 2, a thick line denotes a link with 10Mb bandwidth and 5ms delay, and a thin line denotes a bottleneck link with 5Mb bandwidth and 20ms delay. All TCP connections are bulk data transfer. Either a token bucket or a TSW profiler is used to meter and mark packets from each source node. RIO is used by an interior router to preferentially dropping OUT packets. The two thresholds and the dropping probability used for IN and OUT packets in RIO are 40/70/0.02 and 10/40/0.2, respectively. We refer to the scheme using a TSW profiler and a RIO queue management mechanism as *RIO-TSW*. All the simulations were performed in ns-2 [8], with all TCP connections lasting at least for 20 seconds. The average throughput is calculated at the receiver after the TCP connection reach their steady state.

2.1 The Impact of Profilers/Meters

An effective profiler/meter should have the following properties:

(P1) The average rate of packets marked as IN is approximately equal to the target rate contracted to a flow;

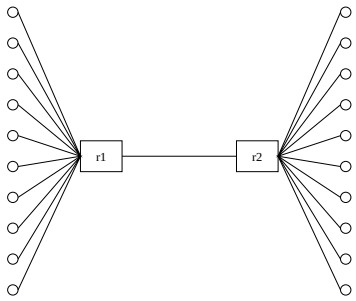

Figure 1: The network topology used in the simulation.

(P2) Packets that are marked as OUT are evenly distributed, regardless of the burstiness nature of TCP traffic, so as to reduce the likelihood of consecutive droppings of OUT packets at core routers; and

(P3) The profiler can accommodate the short term dynamics of TCP traffic and when interfaced with the queue management mechanism, collaboratively fulfill the throughput assurance and fairness requirements of AS.

Problem Associated with the Token Bucket Profiler: A token bucket has two parameters: the size of the token bucket σ and the token generation rate ρ. It is obvious that ρ should be set to the target rate of a connection. The value of σ cannot, however, be easily determined. To demonstrate this, we conduct the following two experiments. In both experiments, there are totally 12 TCP connections, 10 of which are AS connections, and the other 2 are best-effort connections. The ith AS connection requires a target rate of $\lceil (i+1)/2 \rceil$ Mb ($0 \leq i \leq 9$). The bottleneck bandwidth is 33 Mb. All the TCP connections have the same RTTs. The value of σ used for a flow is set to 1/10 of the target rate value of that flow in the first experiment, and 2 Mb in the second experiment. Tables 1–**??** give the goodput, the average rate of IN packets, and the average rate of OUT packets, received at destinations for the two experiments.

Flow	small σ			large σ		
	Total	IN	OUT	Total	IN	OUT
0	1.54	0.71	0.82	0.95	0.93	0.20
1	1.60	0.74	0.86	1.02	0.92	0.86
2	1.84	1.33	0.51	2.20	2.05	0.14
3	2.19	1.47	0.72	2.10	1.98	0.12
4	3.34	1.95	0.38	3.07	2.89	0.18
5	2.65	2.02	0.62	3.01	3.01	0
6	3.19	2.64	0.54	4.07	4.05	0.18
7	3.21	2.65	0.55	4.15	4.09	0.60
8	3.60	3.25	0.34	5.29	5.02	0.27
9	3.67	3.22	0.44	5.32	5.27	0.43
10	0.65	0	0.65	0.56	0	0.56
11	0.53	0	0.53	0.49	0	0.49

Table 1: The performance of token bucket profilers in the case of small values of ρ.

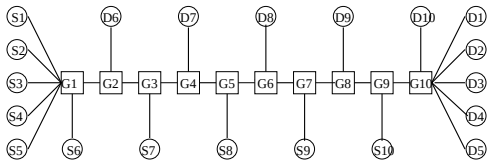

Figure 2: The network topology used in the simulation

As shown in Table 1, the token bucket does not accommodate the burstiness nature of TCP traffic well, when σ is set to a small value (1/10 of the target rate value). Due to token loss when a TCP flow does not transmit at its target rate are lost, the average rate of packets marked as IN is less than the target rate. **(P1)** is violated. On the other hand, in case of large value of σ, token bucket profiler performs reasonabely well. But as shown in Table 1 the OUT packet rates varied a lot among different flows. Since the average rate of OUT packets received at destinations directly reflects how fair the surplus bandwidth is distributed among flows, a large value of σ introduces unfairness. **(P3)** cannot be preserved under this condition. Also, in both cases, some packets in a burst may be consecutively marked as OUT and preferentially dropped at core routers (**(P2)** is violated), causing the congestion window to be repeatedly reduced.

Problem Associated with TSW: TSW has two components: a rate estimator that estimates the packet rate over a time of length *wlen*, and a marker that tags packets as IN (OUT) when the estimated rate is lower than or equal to (greater than) the contracted target rate. There are two approaches to use the TSW as the profiler/meter. In the first approach, the profiler remembers a relatively long past history (*wlen* is large). When a packet arrives, it calculates *avg_rate* as $\frac{avg\_rate \times wlen + packet\_size}{wlen + time\_interval}$. When *avg_rate* exceeds the target rate, packets are marked as OUT with probability $p = \frac{avg\_rate - target\_rate}{avg\_rate}$. The drawback associated with this approach is that it keeps a long past history and cannot reflect well the traffic dynamics of TCP.

In the second approach, the profiler remembers a relatively short history and looks for the peak of a TCP sawtooth when the packet rate exceeds 1.33× the target rate, at which point, the profiler marks packets as OUT with probability p. This approach is more effective to accommodate the TCP traffic dynamics, but as shown in the following experiment, it tends to mark more packets as IN when the actual throughput attainable, r_a, is significantly higher than the target rate. That is, **(P1)** is violated. Setting of the experiment is the same as previous one. Table 2 gives the average rate of IN packets under the use of different values of watermarks (the results of the 2 TCP best-effort packets is zero).

Under the well-subscribed case (i.e., the total contracted profiles/bottleneck bandwidth = 30 Mb/33 Mb = 90%), the watermark of 1.33× the target rate performs better in the sense that the average rate of IN packets is very close to the contracted target rate. (Note that in the case of using 1.1× the target rate as the watermark for the well-subscribed bottleneck link, the average rate of IN packets is less than the contracted target rate for most connections.) On the other hand, in the under-subscribed case (i.e, the total contracted profilers/bottleneck bandwidth = 30 Mb / 50 Mb = 66%), the watermark of 1.1× the target rate outperforms, because TSW with the 1.33× the target rate watermark tends to tag more packets as IN than it should have.

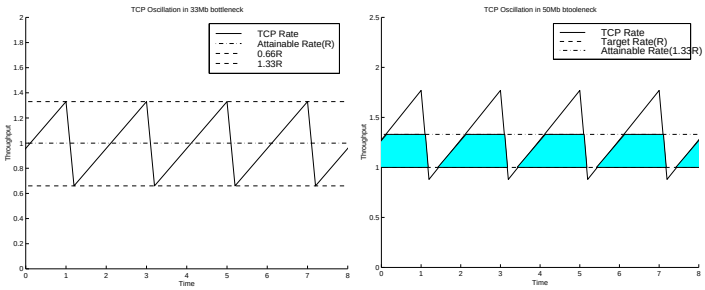

Figure 3: The effect of TCP oscillation on the average rate of packets being marked as IN.

The reason why the performance of TSW is sensitive to the watermark values is as follows. Due to the TCP congestion control mechanism, the instantaneous transmission rate by a TCP connection oscillates between 0.66 and 1.33 of its actual throughput r_a. If r_a is close to the target rate (e.g., under the well-subscribed case, Fig. 3 (a)), 1.33× the target rate serves as a good watermark to tag packets as OUT. Otherwise (Fig. 3 (b)), if r_a is much higher than the target rate, the sawtooth-like curve that depicts the instantaneous TCP transmission rate is shifted up in Fig. 3, allowing more packets to be marked as IN. (The shaded area in Fig. 3 (b) indicates the fraction of extra packets that are marked as IN.)

Flow	1.33 × target rate		1.1 × target rate	
	33Mb	50Mb	33Mb	50Mb
0	1.01	1.22	1.00	1.02
1	1.07	1.20	1.01	1.03
2	2.16	2.46	2.04	2.08
3	2.13	2.42	1.89	2.06
4	3.23	3.47	3.12	3.10
5	3.26	3.30	2.95	3.07
6	4.24	4.83	3.52	3.88
7	4.20	4.35	3.55	3.99
8	5.24	4.98	4.17	4.98
9	5.13	5.77	4.11	4.93

Table 2: The average rate of IN packets under TSW with respect to different watermarks.

Flow	R_T (Mb)	bw = 100 Mb			bw = 33 Mb		
		Ideal	Total	IN	Ideal	Total	IN
0	1	8	6.72	1.26	1.3	1.07	1.01
1	1	8	6.06	1.31	1.3	1.12	1.07
2	2	9	6.53	2.64	2.3	2.23	2.16
3	2	9	7.26	2.59	2.3	2.17	2.13
4	3	10	8.50	3.99	3.3	3.29	3.23
5	3	10	7.64	3.87	3.3	3.33	3.26
6	4	11	9.13	5.31	4.3	4.31	4.24
7	4	11	8.08	5.18	4.3	4.26	4.20
8	5	12	9.45	6.54	5.3	5.28	5.24
9	5	12	8.72	6.56	5.3	5.15	5.13

Table 3: The performance of *RIO-TSW* for TCP connections with respect to different target rates in the cases that the bottleneck link bandwidth is 33Mb and 100Mb, respectively.

2.2 The Interaction between TCP Congestion Control and RIO

Performance of *RIO-TSW* for TCP Connections with Different Target Rates: In this experiment, we study the performance of *RIO-TSW* with respect to different target rates under the well-subscribed and under-subscribed cases. By "well subscribed" ("under subscribed"), we mean that the total contracted profiles take up 90% (60%) of the bottleneck link bandwidth. There are 10 AS TCP connections. The RTT is set to 40 ms for all the TCP connections. Table 3 gives the ideal throughput, the goodput, and the average rate of IN packets accepted at their destinations. Note that the ideal throughput is calculated as the target rate $+\frac{surplusbandwidth}{\#flows}$. Two observations can be made from Table 3: first, although *RIO-TSW* performs reasonably well when the network is well subscribed, it favors connections with small target rates. This phenomenon is especially pronounced when the network is extremely under subscribed. Second, in the case that the network is under-subscribed, TSW tends to mark packets as IN more than it should have as was elaborated on in Section 2.1.

Performance of *RIO-TSW* for TCP Connections with Different RTTs: In the second experiment, there are 2 TCP best-effort connections and 10 TCP AS connections with 5 of them requiring a target rate of 1Mb, and the other 5 requiring a target rate of 5Mb. The RTTs for AS connections of the same target rate varies from 20 ms to 100 ms, respectively. The RTTs for the best-effort connections are set to 40 ms. Fig. 4 gives the simulation results of the goodput versus RTT. As shown in fig. 4, connections with larger values of RTT cannot be assured of its target rate, while those with smaller values of RTT take up more than its target rate. The reason why RIO-TSW cannot fulfil assurance requirement is as follows: due to the TCP congestion control mechnism, a

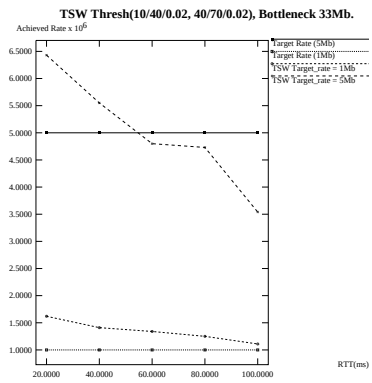

Figure 4: The performance of *RIO-TSW* for TCP connections with different RTTs

connection with larger RTT or larger target rate, tends to take more regain its original window size after a packet loss. Similar findings are also reported in [?].

3 The Enhanced Time Sliding Window Profiler

```
/* Upon arrival of a packet from flow i */
1.  estimate the long-term transmission rate R_l;
2.  estimate the short-term sending rate R_s;
3.  if (R_l − R_s) > threshold
4.      watermark ← watermark +δ;
5.  else if (R_s − R_l) > threshold
6.      watermark ← watermark −δ;
7.  if (R_s < r^in × watermark)
8.      mark the packet as IN;
9.  else {
        /* calculate the dropping probability for flow i */
10.     p_i ← (R_s − r^in × watermark)/R_s;
11.     mark the packet as OUT with probability p_i;
12. }
```

Figure 5: The procedure taken to update the watermark and to mark packets in ETSW.

We now propose an enhanced version of TSW, called *ETSW*, that retains the nice features of both the short-term and long-term tsw profilers, while eliminating their drawbacks. Fig. 3 outlines the procedure taken by ETSW to update the watermark and to mark packets. Instead of using one fixed value as the watermark, ETSW adjusts its watermark between 1.0 and 1.33 based on both the long-term and short-term average rates. Both *threshold* and the increment/decrement step δ are tunable parameters. (In our experiment, $\delta = 0.01$) Table 3 gives the simulation results of ETSW under the well- and under-subscribed cases for the same simulation setting as in Table 2. By comparing Table 3 and Table 2, one can see that ETSW outperforms both in the well- and under-subscribed cases in the sense that the average

Flow	bw = 33Mb			bw = 50Mb		
	Total	IN	OUT	Total	IN	OUT
0	1.22	0.95	0.26	1.70	0.98	0.71
1	1.23	0.95	0.27	1.81	0.97	0.83
2	2.23	1.99	0.23	3.30	1.95	1.34
3	2.20	1.97	0.22	3.33	2.02	1.31
4	3.17	3.00	0.17	3.90	2.96	0.94
5	3.17	2.98	0.19	4.45	3.01	1.44
6	4.10	3.98	0.12	5.28	3.86	1.42
7	4.12	3.97	0.14	5.19	4.00	1.20
8	5.13	4.99	0.13	5.73	5.02	0.70
9	5.03	4.92	0.11	6.70	5.04	1.66
10	0.16	0.00	1.68	0.56	0.00	0.56
11	0.15	0.00	1.58	1.68	0.00	1.68

Table 4: The performance of ETSW under the well- and under-subscribed cases.

rate of packets marked as IN for a connection is approximately equal to its contracted target rate.

4 The (r^{in}, RTT)-Adaptive Scheme

In this and next sections, we discuss two schemes that alleviate the aforementioned undesirable coupling effect of TCP congestion scheme and RIO queue mangement mechniasm, and improve the throughput assurance and fairness for AS. The first strategy, which we call the (r^{in}, RTT)-adaptive scheme, intends to alleviate the effects of r^{in} and RTT on the throughput by figuring them in the calculation of the dropping probability, p_i^{out}, for flow i used in RIO. The second strategy, which we call the *dynamic RIO* scheme, monitors, without keeping the per-flow queue structure, the number of OUT packets of a connection received at each core router, and punishes connections that send more OUT packets than their fair share.

There are two major features associated with the (r^{in}, RTT)-adaptive scheme. First, each output interface of a core router is equipped with a three queue structure: one queue is for packets that require the premium service, another is for packets that require the assured service, and the other is for the best-effort traffic. The three queues are served by simple priority, premium packets first. In the case that there is no premium packet present, if the first packet in the AS queue is an IN packet, it is served; otherwise, with probability $\frac{n_a}{n_a+n_b}$ the packet is served, and with probability $\frac{n_b}{n_a+n_b}$ the first packet in the best effort queue is served, where n_a and n_b are the number of active AS and best-effort flows at the core router, and can be on-line estimated using the zombie list mechanism reported in [6]. The isolation of AS flows from best-effort flows by enqueueing their respective packets into two separate queues prevents non-adaptive best-effort UDP flows from consuming a large portion of buffers (and hence the output bandwidth) at the expense of deteriorating the throughput assurance of adaptive TCP AS flows.

Second, the dropping probability used in the random early drop mode is now dynamically adjusted based on r^{in} and RTT. In RIO, the dropping probability, p_i^{out}, is given by:

$$p^{out} = p_{max}^{out} \times \frac{avg\_queue - minth\_out}{maxth\_out - minth\_out}, \qquad (4.1)$$

for $minth\_out \leq avg\_queue \leq maxth\_out$, where p_{max}^{out} is same for all flows.

In the (r^{in}, RTT)-adaptive scheme, Eq. (4.1) is still valid, but p_{max}^{out} is now configured as a function of r^{in} and RTT and may change for individual flows. Conceptually, we increase p_{max}^{out} for flows with smaller values of r^{in} and RTT so as to compensate the coupling effect of TCP congestion control dynamics and RIO queue management.

First, we define the following terms:

(i) r_i^{in}, r_i^{out}: the rates at which flow i sends its IN and OUT packets, respectively;

(ii) p_i^{in}, p_i^{out}: the (steady state) probability of IN and OUT packet dropping probability of flow i, respectively at a core router. If the queue management mechanism is well provisioned, $p_i^{in} \cong 0$;

(iii) $R_i = r_i^{in} + r_i^{out}$: the total sending rate of flow i;

(iv) p_i: the (steady state) packet dropping probability of flow i, i.e., $p_i = \frac{r_i^{in}}{r_i^{in}+r_i^{out}} \cdot p_i^{in} + \frac{r_i^{out}}{r_i^{in}+r_i^{out}} \cdot p_i^{out}$;

(v) T_i: the sustained throughput of flow i. If the queue management mechanism is well provisioned, $T_i \cong r_i^{in} + r_i^{out}(1 - p_i^{out})$;

(vi) τ_i: the round trip time of flow i;

(vii) d_i: the packet dropping rate of flow i.

We consider two flows that share a bottleneck link. The ratio of packet dropping rates between the two flows can be expressed as

$$\begin{aligned} d_1 : d_2 &= R_1 \cdot p_1 : R_2 \cdot p_2 \\ &\cong T_1 \cdot p_1 : T_2 \cdot p_2 \quad \text{under } d_i << T_i \\ &\cong \frac{1}{T_1 \cdot \tau_1^2} : \frac{1}{T_2 \cdot \tau_2^2} \quad T_i \propto \frac{1}{\tau_i \sqrt{p_i}} \\ &\cong \frac{[r_2^{in} + r_2^{out}(1 - p_2^{out})] \cdot \tau_2^2}{[r_1^{in} + r_1^{out}(1 - p_1^{out})] \cdot \tau_1^2}. \end{aligned} \qquad (4.2)$$

On the other hand, the ratio of packet dropping rates between the two flows can also be expressed as

$$\begin{aligned} d_1 : d_2 &= (r_1^{in} p_1^{in} + r_1^{out} p_1^{out}) : (r_2^{in} \cdot p_2^{in} + r_2^{out} \cdot p_2^{out}) \\ &\cong \frac{r_1^{out} p_1^{out}}{r_2^{out} p_2^{out}}, \quad \text{assuming that } p_1^{in} \cong 0 \end{aligned} \qquad (4.3)$$

Let $n_f = n_a + n_b$ denote the number of active AS and best-effort flows at the core router, B the available bandwidth for AS and best-effort flows, and $S \triangleq (B - \sum r_i^{in})/n_f$ the surplus bandwidth to be evenly shared per active flow. To enforce fairness, the rate of OUT packets received at destinations should be equal between the two connections:

$$r_1^{out} \cdot (1 - p_1^{out}) = r_2^{out} \cdot (1 - p_2^{out}) = S. \quad (4.4)$$

Hence,

$$d_1 : d_2 = \frac{p_1^{out}}{1 - p_1^{out}} : \frac{p_2^{out}}{1 - p_2^{out}}. \quad (4.5)$$

Equating Eqs. (4.2) and (4.5), we have

$$\frac{r_i^{in} \cdot p_i^{out} \cdot \tau_i^2}{1 - p_i^{out}} + p_i^{out} \cdot r_i^{out} \cdot \tau_i^2 = const. \quad (4.6)$$

With Eq. (4.4), Eq. (4.6) can be re-written as

$$\frac{p_i^{out} \cdot \tau_i^2}{1 - p_i^{out}}(r_i^{in} + S) = const. \quad (4.7)$$

Eq. (4.7) along with the boundary condition

$$(\sum_i r_i^{in} \tau_i^2) \cdot \frac{p_{max}^{out}}{1 - p_{max}^{out}} = \sum_i \frac{r_i^{in} \cdot p_i^{out} \cdot \tau_i^2}{1 - p_i^{out}}, \quad (4.8)$$

where p_{max}^{out} is the default value used in RIO, can be used to numerically calculate p_i^{out} if the parameters r_i^{in}, τ_i, and S are known upon arrival of an OUT packet from flow i. In fact, all possible values of p_i^{out} can be calculated off-line using Eqs. (4.7)–(4.8) and stored in a table indexed by r_i^{in}, τ_i, and S. Each packet carries r_i^{in} and τ_i in its header. S can be on-line estimated by the core router using the zombie list mechanism in [6]. Upon arrival of an OUT packet, a core router uses the parameters as indices, does a table lookup to determine the value of p_i^{out}, and uses the value found as p_{max}^{out} in Eq. (4.1).

5 The DRIO Scheme

As will be discussed in Section 6, the (r^{in}, RTT)-adaptive scheme does not perform as well as expected under the case AS flows incur different values of RTTs. In this section, we propose an alternative strategy, called *DRIO*, in which a core router monitors, with the use of a single *history list*, the OUT packets sent by connections and preferentially drops OUT packets sent by connections that have sent more OUT packets than their fair share. *DRIO* achieves per-flow monitoring without requiring the per-flow queue structure, and also adapts well in the case that multiple bottleneck links exist.

Fig. 5 outlines the operations of DRIO. Each core router keeps, for each of its outgoing interfaces, a history list of IDs for K flows whose OUT packets are recently seen. The history list starts out empty.

```
/* Upon arrival of an OUT packet pkt from flow i */
1. if (avg_queue_length > maxth_out)
2.     drop(pkt);
3. else if (avg_queue_length > minth_out) {
4.     /* DRIO takes effect */
5.     if (!full(history_list))
6.         insert(pkt.flow_id, &history_list);
7.     else {
       /* randomly pick up two elements from history_list
          and compare pkt.flow_id against them */
8.         num_hit ← history_comparison(pkt.flow_id, history_list);
9.         if (num_hit == 2) { /* double hit */
10.            drop(pkt);
11.            p_i^out ← p_i^out + δ;
12.            if (p_i^out > 1.0) p_i^out ← 1;
13.        }
14.        else if (num_hit == 1) /* one hit and one miss */
               /* with probability p_i^out drop the packet */
15.            if (rand()≥ p_i^out) drop(pkt);
16.            else
17.                insert(pkt, &pkt_queue);
18.        else { /* double miss */
19.            replace one selected element with pkt.flow_id;
20.            insert(pkt, &pkt_queue);
21.            if (rand() ≥ q) {
       /* q is universally used by all the connections */
22.                p_i^out ← p_i^out − δ;
23.                if (p_i^out < 0.0) p_i^out ← 0;
24.            }
25.        }
26.    } /* end of the case of a full history_list */
27. } /* else if */
28. else /* avg_queue_length ≤ minth_out */
29.    insert(pkt, &pkt_queue);
```

Figure 6: The operations of DRIO.

Upon receipt of an OUT packet, if the average queue length falls *between* the two RIO watermarks, *minth_out* and *maxth_out*, DRIO takes effect: if the history list is not full, the flow id of the OUT packet is added to the history list; otherwise, two elements are randomly selected from the history list and compared against the flow id of the OUT packet. We say that a hit (miss) occurs if the flow id of the OUT packet is (not) the same as that selected from the history list. There are three possible outcomes for the comparison: a double hit, a double miss, or a combination of one hit and one miss.

In the case of a double hit, the packet is dropped and the probability, p_i^{out}, of dropping OUT packets for this connection is increased by a fixed amount, δ, where δ is a tunable parameter. In the case of one hit and one miss, the packet is dropped with probability p_i^{out}. In the case of a double miss, one of the two elements selected from the history list is replaced with the flow id of the packet, the packet is enqueued , and p_i^{out} is decreased by a fixed amount, δ, with probability q, where q is a parameter used by all the connections whose value is yet to be determined (Section 5.1).

Note that to implement DRIO, we have to keep track of p_i^{out}'s for active flows. To this end, we maintain a separate list, called the probability list, each entry of which contains (i) the flow id of an active flow, (ii) the value of p_i^{out} used for the flow, and (iii) the time stamp when the entry was last updated. In the case that the dropping probability of a flow i has to be adjusted, the probability list is searched. If no entry can be found with a matched flow id, flow i is assumed to be a newly active flow and starts with a default value; otherwise, the corresponding p_i^{out} value is used. The outdated entries in the probability list are periodically purged based on timestamp value.

The rationale behind the DRIO operations is as follows. Since buffer occupancy in the history list depends on the rate at which each flow sends its OUT packets, the probability that a hit occurs upon arrival of an OUT packet from flow i (in the single bottleneck link case) can be approximated as

$$p_h = \frac{r_i^{out}}{\sum_j r_j^{out}}. \tag{5.1}$$

If flow i tends to send more OUT packets, r_i^{out} (and hence p_h) is large. As a result, the number of hits serves as a good indication of whether or not a flow has sent excessive OUT packets. DRIO can be readily extended to the general case in which K elements are randomly picked up from the history list for comparison. Simulation results show that a scheme with a single comparison ($K = 1$) is not as sensitive in punishing aggressive flows as DRIO and the gain in increasing the number of comparisons beyond 2 is marginal. To prevent p_i^{out} from being too small, p_i^{out} only decreased with probability q when double miss occurs. The value of q is critical to the Section 5.1.

5.1 Analysis on DRIO

The analysis serves two purposes: (1) determination of the probability q; (2) study of how sensitive DRIO is in reacting to aggressive flows that send excessive OUT packets and achieving fairness.

Discrete-Time Markov Chain: For ease of analysis, we consider the case of a single bottleneck link. We model the system evolution of a flow i under DRIO as an embedded discrete-time Markov chain (DTMC). Let the packet dropping probability, p_i^{out}, of flow i, be denoted as $p_i^{out} = k \times \delta$, where $0 \le k \le N \triangleq \lfloor 1.0/\delta \rfloor$. The system is observed upon every packet arrival when DRIO takes effect and the state of the DTMC is defined as k. The probability of a double hit, p_{2h}, and the probability of a double miss, p_{2m}, can be expressed, respectively, as

$$p_{2h} = (\frac{r_i^{out}}{\sum_j r_j^{out}})^2, \tag{5.2}$$

$$p_{2m} = (1 - \frac{r_i^{out}}{\sum_j r_j^{out}})^2. \tag{5.3}$$

The state transition probability, P_{ij}, for the DTMC, is thus

$$P_{ij} = \begin{cases} p_{2h} & j = i+1, \\ 1 - p_{2h} - p_{2m} \cdot q & j = i, \\ p_{2m} \cdot q & j = i-1. \end{cases} \tag{5.4}$$

Let the steady state probability that the DTMC is in state k be denoted as $\pi(k)$, and $\pi = [\pi(0), \pi(1), \ldots, \pi(N)]$ and $\mathbf{P} = [P_{ij}]$. Solving $\pi = \pi\ \mathbf{P}$ and $\sum_k \pi(k) = 1$, we have

$$\pi(k) = \frac{\rho^k}{\sum_{i=0}^{N} \rho^i}, \quad 0 \le k \le N, \tag{5.5}$$

where

$$\rho = \frac{p_{2h}}{p_{2m} \cdot q} = \frac{(r_i^{out})^2}{(\sum_{j \ne i} r_j^{out})^2 \cdot q}. \tag{5.6}$$

The average probability of dropping OUT packets for flow i is thus

$$\overline{p_i^{out}} = \delta \times \sum_{k=0}^{N} k \cdot \pi(k) = \delta \times \frac{\sum_{k=0}^{N} k \cdot \rho^k}{\sum_{i=0}^{N} \rho^i}. \tag{5.7}$$

Determination of q: For a given desirable value of $\overline{p_i^{out}}$, the values of ρ can be determined from Eq. (5.7). Denote the value of ρ thus obtained as $\hat{\rho}$. Then, $q = (\frac{r_i^{out}}{\sum_{j \ne i} r_j^{out}})^2 \cdot \frac{1}{\hat{\rho}}$ in DRIO. Now to enforce fairness, we use in DRIO the value of q obtained under the assumption that all the flows send their OUT packets at the same rate ($r_i^{out} = r_j^{out}$):

$$q = \frac{1}{(n_f - 1)^2 \cdot \hat{\rho}}, \tag{5.8}$$

where n_f is the number of active flows and can be estimated using the zombie list mechanism proposed in [6].

Sensitivity study of DRIO: If a flow i tends to send more OUT packets $\ell (> 1)$ times than the other flows, the value of ρ in its corresponding DTMC becomes

$$\rho \text{ for flow } i = (\frac{(n-1)r_i^{out}}{\sum_{j \ne i} r_j^{out}})^2 \cdot \hat{\rho} = \ell^2 \hat{\rho}. \tag{5.9}$$

This implies that the packet dropping probability of flow i is ℓ^2 times more likely to be increased than to be decreased as compared to the other flows. That is, DRIO penalizes flow i by dropping more of its OUT packets. To demonstrate how sensitive DRIO is in reacting to aggressive flows that send OUT packets more than their fair share, we depict in Fig. 7 $\overline{p_i^{out}}$ as a function of $r_i^{out} / \sum_j r_j^{out}$ under DRIO. (Also shown in Fig. 7 (b) is $\overline{p_i^{out}}$ under the schemes with a single comparison and

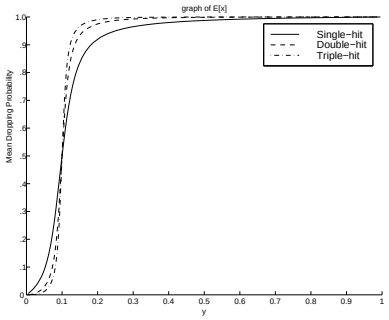

Figure 7: Mean of the dropping probability p_i^{out} vs. the probability of a hit $p_h = \frac{r_i^{out}}{\sum_j r_j^{out}}$,$n_f = 10$.

with $K = 3$ comparisons, respectively.) Without loss of generality, $\hat{\rho}$ is set to be one. The reflection point occurs at $\frac{r_i^{out}}{\sum_j r_j^{out}} = 1/n_f$ (Fig. 7). When flow i sends OUT packets more than its fair share, $\frac{r_i^{out}}{\sum_j r_j^{out}} > 1/n_f$ and $\overline{p_i^{out}}$ increases rapidly to 1 with $\frac{r_i^{out}}{\sum_j r_j^{out}}$ (and vice versa) under DRIO. The sensitivity level of DRIO in increasing the packet dropping probability increases as the number of active flows increases. There is a gain in the sensitivity level using more comparisons, but the gain beyond $K = 2$ is marginal.

6 Simulation Results

We have implemented in ns-2 [8] the proposed (r^{in}, RTT)-adaptive-ETSW and DRIO-ETSW schemes, along with the 2-window TCP scheme [3], the three color-marker scheme [4, **?**], and the CSFQ scheme [**?**], and conducted a simulation study to validate the proposed design and compare the performance of the proposed and existing schemes under a vairety of network topologies and traffice sources. The two thresholds and the dropping probability used for IN and OUT packets in RIO-TSW are 40/70/0.02 and 10/40/0.2, respectively. Each data point is the result averaged over 5 simulation runs.

6.1 Comparison among proposed schemes

Comparison with respect to different target rates: Fig. 8 gives the performance comparison among Adaptive-ETSW, DRIO-ETSW and RIO-TSW with respect to different target rates in the case that the bottleneck link is 50Mb The simulation setting is the same as that in Table 2. The ideal value is calculated as the target rate $+\frac{\text{surplus bandwidth}}{\text{the total number of flows}}$. Adaptive-ETSW performs better than RIO-TSW in terms of fairness in both cases (the curve associated with Adaptive-ETSW is in more parallel with the ideal curve in Fig. 8). The degree of improvement is even more pronounced under DRIO-ETSW.

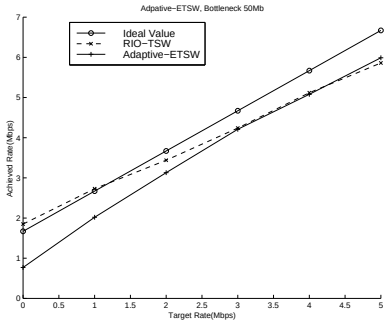

Figure 8: The comparison among Adaptive-ETSW, DRIO-ETSW and RIO-TSW with respect to different target rates in the case the bottleneck link is 50Mb

Comparison with respect to RTTs: Fig. 9 compares Adaptive-ETSW, DRIO-ETSW, and RIO-TSW with respect to RTTs in the case that the bottleneck link is 33Mb , respectively. The simulation setting is the same as that in Fig. 4. RIO-TSW is greatly affected by RTT, and cannot fulfill the throughput assurance requirement for 5-Mb flows (1-Mb flows) when RTT $\geq$ 70 ms (RTT $\geq$ 85 ms). Adaptive-ETSW performs better than RIO-TSW, but is still affected by RTT. We believe this is because the vale of p_i^{out} calculated using Eqs. (4.7)–(4.8) does not differ significantly enough under different RTT values (the values of p_i^{out} usually differ only in the second/third digit after the decimal point). DRIO-ETSW is rather insensitive to the values of RTT, and performs significantly better both in terms of throughput assurance and fairness.

Comparison in the existence of non-responsive UDP flows: To study how the three schemes perform in the existence of UDP flows, we simulate 12 connections, 10 of which are TCP AS connections, with target rates ranging from 1 Mb to 5 Mb, respectively, and the others are UDP CBR connections with the sending rate 10 Mb. The RTT is set to 40 ms for all the connections. Table 5 gives the simulation results in the existence

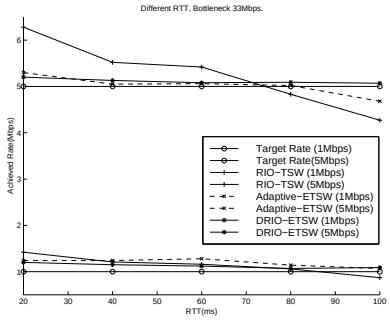

Figure 9: Performance comparison among Adaptive-ETSW, DRIO-ETSW, and RIO-TSW with respect to RTTs in the case that the bottleneck link is 33Mb

Flow	r^{in} (Mb)	RIO-TSW		Adaptive		DRIO	
		Total	IN	Total	IN	Total	IN
TCP 0	1	1.42	0.94	1.94	0.98	2.20	0.99
TCP 1	1	1.44	0.93	2.34	0.96	2.20	1.00
TCP 2	2	2.23	1.81	3.28	2.01	2.99	1.97
TCP 3	2	2.12	1.77	3.27	1.94	2.74	1.96
TCP 4	3	2.92	2.68	4.49	3.07	3.88	2.97
TCP 5	3	3.08	2.82	4.46	2.97	3.88	2.97
TCP 6	4	3.75	3.54	5.61	4.10	4.81	3.97
TCP 7	4	3.88	3.64	5.27	4.02	4.74	3.99
TCP 8	5	4.77	4.59	6.25	4.95	5.65	4.93
TCP 9	5	4.6965	4.51	6.24	4.97	5.59	4.92
UDP 0	0	9.60	0	1.26	0	5.50	0
UDP 1	0	9.58	0	1.22	0	5.53	0

Table 5: The performance of RIO-TSW, Adaptive-ETSW, and DRIO- ETSW in the existence of UDP flows.

of non-responsive UDP CBR connections. RIO-TSW is greatly affected by the existence of UDP flows, and some of the connections cannot attain their target rates. The reason is as follows. The surplus bandwidth to be evenly shared among connections in the 50 Mb bottleneck link case is $(50-30)/12 = 1.66$ Mb, but each of the two UDP flows captures more than 9.58 Mb. Consequently, the maximum threshold, *maxth_out*, for OUT packets is constantly reached, and most of the OUT packets of TCP AS connections get dropped (see the two columns under RIO-TSW in Table 5). As a result, the TCP connections alternate between slow-start and congestion avoidance phases, never going through the fast-recovery phase, and hence cannot reach their target rates.

DRIO-ETSW can detect the flows that send excessive OUT packets and preferentially drop their OUT packets, and hence UDP flows cannot capture the surplus bandwidth as aggressively as in RIO-TSW. Some of the OUT packets of TCP AS connections may make their way through the bottleneck link. However, DRIO-ETSW still cannot totally eliminate the effect of non-responsive UDP flows. Adaptive-ETSW, on the other hand, is able to isolate UDP packets from AS packets because of the three-queue architecture used at the core router, and hence has the best performance.

Comparison under multiple bottleneck links: We also consider the performance of proposed schemes under the multiple bottleneck link topology (Fig. 2). There are 5 TCP AS connections, $S_i \rightarrow D_i$, $1 \leq i \leq 5$ along the long path, each of which requests a target rate of 0.2 Mb. On the long path, there are another 5 interfering TCP connections, $S_i \rightarrow D_i$, $6 \leq i \leq 10$, each of which requests a target rate of 1.0 Mb. (Note that the target-rate of the aggregated traffic along the long path is equal to that of the interfering traffic.) All TCP connections run for 100 ms.

Fig. 10 gives the simulation results under the multiple bottleneck link case. We observe similar trends in

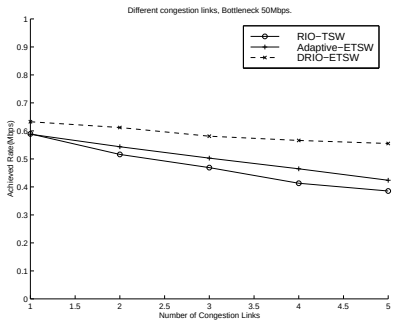

Figure 10: The performance of *RIO-TSW*, *Adaptive-ETSW* and *DRIO-ETSW* under multiple bottleneck topology

the single bottleneck link case, except that in the performance gap between DRIO-ETSW and RIO-TSW is even more pronounced.

In RIO-TSW, as the number of congested bottlenecks increases, the goodput of connections along the long path decreases. In contrast, DRIO-ETSW dynamically monitors the packet arrival pattern at each core router along the path and punishes the connections that potentially send more OUT packets and consume more bandwidth that its fair share. As a result, it is not susceptible to the number of bottleneck links.

6.2 Comparison with existing schemes

In this subsection, we compare the performance of DRIO-ETSW, Adaptive-ETSW, (single-rate) three-color marker, two-window TCP, and CSFQ. Note that the three-color marker is only a profiler, we use triple-RED gateway (or RED with Green, Yellow, and Red) as the queue management mechnism which is a natually extension of RIO.

For the three-color marker (single rate) coupled with the triple-RED queue management mechanism, the parameters for the profiler are set as follows: CIR = the target rate of a flow, $CBS = 3.0$ Mb, and $EBS = 1.0$ Mb. The parameters for the triple-RED queue management mechanism are set as follows: (*minth_green*, *maxth_green*, p_{max}^{green}) = (40, 70, 0.002), (*minth_yellow*, *maxth_yellow*, p_{max}^{yellow}) = (40, 70, 0.002), and (*minth_red*, *maxth_red*, p_{max}^{red}) = (10, 40, 0.02). For CSFQ, the averaging constant K (used in estimating the flow rate), K_α (used in estimating the fair rate), and K_c (used in making the decision of whether or not a link is congested) are all set to 100 ms. For two-window TCP, the token bucket size is set to 100 ms $\times$ the token generation rate.

In the simulation, all 10 TCP flows have the same target rates/weights, but their RTTs vary from 20 ms to 100 ms.

Several observations can be made from the fig. 12. First, CSFQ and two-window TCp are sensitive to varia-

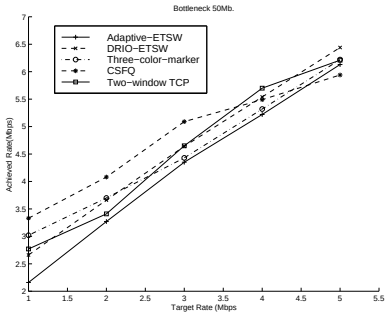

Figure 11: Performance comparison of Adaptive-ETSW, DRIO-ETSW, three-color marker, two-window TCP, and CSFQ with respect to different target rates.

tion of RTT, becaue both schemes don not consider how to alleviate the effect of TCP window dynamics on the throughput assurance. Second, three-color marker coupled with the triple-RED queue management mechanism performs better than TSW-RIO, but the improvement is not significant. Third, in nearly all the simulation runs, DRIO-ETSW outperforms the other schemes in terms of both throughput assurance and fairness.

7 Conclusion

In this paper, we have studied why the current AS architecture cannot provide throughput assurance and fairness. Based on the findings of the simulation study, we then propose an enhanced version of TSW, called *ETSW*, and devise, based on analytical reasoning, two enhanced versions of the RIO queue management mechanism, called, respectively, the (r, RTT)-*adaptive* algorithm and the *dynamic RIO* (DRIO) algorithm. We have also implemented in ns-2 the proposed mechanisms, along with two-window TCP [3], three-color marker [4, ?], and CSFQ [?], and examine their behavior under a variety of network topologies and traffic sources. The simulation results indicate that both DRIO and (r, RTT)-adaptive algorithms, when combined with ETSW, do fulfill more satisfyingly the throughput assurance and fairness requirements. The simulation results also indicate the proposed schemes outperform all other existing schemes.

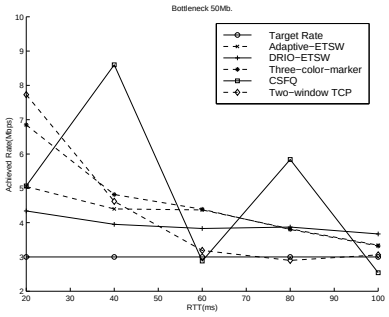

Figure 12: Performance comparison of Adaptive-ETSW, DRIO-ETSW, three-color marker, two-window TCP, and CSFQ with respect to different RTTs.

References

[1] W. Lin, R. Zheng, C-J. Hou. Technical Report. *http://eewww.eng.ohio-state.edu.cn/ jhou*

[2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. RFC 2475, December 1998.

[3] W. Feng, Dilip D. Kandlur, D. Saha, and Kang G. Shin. Understanding TCP dynamics in an integrated services Internet. *Proc. of the Int'l Workshop on NOSSDAV*, May 1997.

[4] J. Heinanen and R. Guerin. A single rate three color marker. Work in progress, IETF draft, March 1999.

[5] K. Nichols, V. Jacobson, and L. Zhang. A two-bit differentiated services architecture for the Internet. Work in progress, IETF draft, Nov. 1997.

[6] T. J. Ott, T. V. Lakshman, and L. H. Wong. SRED: stabilized RED. *Proc. of IEEE INFOCOM'99*, New York, NY, March 1999.

[7] I. Stoica, S. Shenker, and H. Zhang. Core-stateless fair queuing: achieving approximation fair bandwidth allocations in high speed networks. *Proc. of ACM SIGCOMM'98*, October 1998, pp. 118-130.

[8] UCB, LBNL, VINT Network Simulator. *http://www-mash.cs.berkeley.edu/ns/*.