Reasons not to deploy RED

Martin May⋄, Jean Bolot⋄, Christophe Diot*, and Bryan Lyles*

⋄ INRIA
mmay,bolot@sophia.inria.fr

* SPRINT Labs
cdiot,lyles@sprintlabs.com

Abstract— **In this paper we question the benefits of RED by using a testbed made of two CISCO 7500 routers and up to 16 PCs to observe RED performance under a traffic load made of FTP transfers, together with HTTP traffic and non responsive UDP flows. The main results we found were, first, that RED with small buffers does *not* improve significantly the performance of the network, in particular the overall throughput is smaller than with Tail Drop and the difference in delay is not significant. Second, parameter tuning in RED remains an inexact science, but has no big impact on the end-to-end performance. We argue that RED deployment is not straight forward, and we strongly recommend more research with realistic network settings to develop a full quantitative understanding of RED. Nevertheless, RED allows us to control the queue size with large buffers.**

I. Introduction

To the question: "would you implement RED on your network?", our preferred carrier answered with some kind of surprise: "why? why would I drop perfectly good packets when there is no obvious reason to do so, and why would I change a really simple mechanism (i.e.tail drop) that works perfectly for a more complex mechanism for which I have no proof it works better".

Consequently, we decided to take a public implementation of RED (CISCO IOS 12.0 [7]) and to run some experiments on a local testbed. We use Chariot 2.2 [4] load generator to simulate a classic Internet traffic with a decent number of connections. Our experiments highlight the impact of network traffic conditions, router settings, and RED parameter choice on the end2end transmission performance with RED.

Parameter choice in RED seems to be hard since the inventors periodically changes their recommended RED parameters [2]. Even more, Van Jacobson recently claimed, that any parameter setting (as long as the control law is monotone, non-deceasing and covers the full range of 0 to 100% drop rate) will work and will improve the system performance [5].

Many of the intuitions which have driven this work have derived from previous experiences with the generation of congestion control feedback, for example, the ABR work in the ATM Forum. In particular, we note that the Forum found that systems which generate feedback based on the bulk behavior of traffic have parameters which are sensitive to the exact network configuration and traffic mix. In the case of the development of ABR this led to a long period of parameter optimizations, and ultimately to the development of systems which gave per flow feedback.

II. Experimental platform

Significant numbers of papers on RED have been published based on simulation studies. While simulation is a core tool for network protocol investigation, the traffic generated by most of the simulators is quite different from real network traffic (most simulations use infinite greedy TCP source, RTT is constant, and simulation limits the number of connections). Therefore, in this study we used Chariot, a network load generator to generate, manage and synchronize a whole set of traffic connections on different endpoints (PCs running MS Windows NT4.0) where the traffic more closely approximated real network traffic. Chariot simulates e.g., FTP connections (with the setup phase and for different file sizes), file server traffic, HTTP traffic from and towards a web server, or real-time audio and video traffic. All these transfers were using the real TCP or UDP implementations on the endpoints.

The setup we used is illustrated in Figure 1.

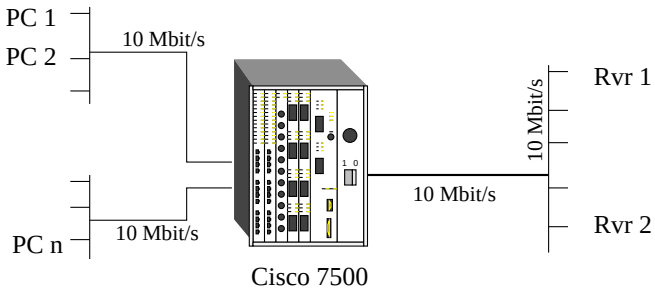

Fig. 1. Testbed Setup

Our traffic was made of 80% of TCP connections and 20% of UDP flows ... 60% of the TCP traffic was generated by FTP sources sending files of variable sizes about 100000 bytes. The missing 40% is HTTP traffic for text (small transfers) and pictures (large transfers).

III. Parameter setting for RED with one router

In this section, we evaluate performance of RED and we compare these results to those obtained when using tail drop with the same network traffic. Unfortunately, [3] offers little guidance on how to set configuration parameters. We found the best guidance in the CISCO IOS online documentation:

"Note: The default WRED parameter values are based on the best available data. CISCO recommends that you do not change the parameters from their default values unless you have determined that your applications would benefit from the changed values. "

Thus, we tried to understand how to tune RED parameters for our applications.

Before we describe our experimental observation, we provide a short reminder on RED, and the correspondence between RED IOS parameters and parameters found in the RED manifesto [1].

RED randomly drops or marks arriving packets when the average queue length exceeds a minimum threshold. The drop probability increases with increasing average queue length up to a maximal dropping probability. When the average queue size reaches an upper threshold all packets are dropped. The average queue size is calculated with an exponential weighted moving average, where a variable α defines the weight for the past queue size values. The CISCO IOS allows to set the following 4 parameters: the minimum threshold min_{th}, the maximum threshold max_{th}, the maximum drop probability max_p, and the averaging parameter α.

In the following we examine the end2end performance of our network with varying RED parameters. Therefore, we used different RED setups by varying max_p, the size of the dropping interval ($max_{th} - min_{th}$), the buffer size, and the averaging parameter α. To evaluate the performance of RED we measured 3 values of interest: the average throughput ($Throughput$), the number of bytes sent ($BytesSent * 1000$), and the percentage of UDP packets lost ($\%UDPdrop$).

A. The maximum drop probability max_p

This parameter should be chosen *reasonably* so that sufficient drops occur between min_{th} and max_{th}. [3] as well as [7] recommends a value no larger than 0.1. We used values from 1/100 to 1 and compared those with a simple tail drop buffer management.

In table I we used a *default* RED setting with an outgoing buffer size of 40 packets (default value for all CISCO interface cards) and set $min - th = 10$ and $max_{th} = 30$. The value for the averaging of the queue size is set to $\alpha = 0.002$.

max_p	Throughput	Bytes Sent	% UDP drop
1/100	8.471	626,326	9.110
1/10	8.304	623,830	9.244
1	8.294	618,534	9.713

TABLE I. Performance for a buffer size of 40 packets and varying max_p

Our next experiment was similar, but this time we used larger buffers in the router. In particular, we set $min_{th} = 30$ and $max_{th} = 130$ for a buffer size of 200 packets. For the small buffer the increase of the drop

max_p	Throughput	Bytes Sent	% UDP drop
1/100	8.474	632,347	1.529
1/10	8.337	622,844	1.693
1	8.476	628,231	2.180
TD	8.395	629,244	5.431

TABLE II. Performance for a buffer size of 200 packets and varying max_p

probability results in a minor increase of the TCP performance without changing the UDP drop rate. The overall router throughput is not changing significantly. With the larger buffer the observations are different. TCP performance does not seem to be a simple function of the setting of max_p. Interestingly, the UDP drop rate is higher with tail drop. So it appears that Tail Drop penalizes non-responsive traffic to a greater extent than RED.

B. Varying the size of the dropping interval ($max_{th} - min_{th}$)

In the second test, we varied the size of the dropping interval. This is to avoid the deterministic dropping when the maximum threshold is reached. Floyd recommends $max_{th} = 2 * min_{th}$ in [3].

Again we compared the performance with a small (40 packets) and a large (200 packets) buffer. For both tests we set the drop probability to a constant $max_p = 1/10$.

Interval	Throughput	Bytes Sent	% UDP drop
5 to 10	8.347	620,423	9.921
5 to 20	8.472	623,331	9.684
5 to 30	8.384	624,533	9.306

TABLE III. Performance for a buffer size of 200 packets and varying max_p

The same results for a buffer size of 200 packets a presented table IV. Again, we observe that a variation of the dropping interval, i.e. the difference between the max_{th} and the min_{th} does not influence the system performance for TCP as well as for UDP traffic. As

Interval	Throughput	Bytes Sent	% UDP drop
20 to 50	8.501	632,736	2.000
20 to 100	8.480	632,942	2.013
20 to 150	8.470	631,543	2.685
TD	8.395	629,244	5.431

TABLE IV. Performance for a buffer size of 200 packets and varying max_p

Buffer Size	Throughput	Bytes Sent	% UDP drop
RED 40	8.331	623,630	9.429
RED 100	8.408	629,335	6.407
RED 150	8.579	632,037	4.825
RED 200	8.480	633,037	3.325
TD $b = 50$	8.253	622,307	12.495
TD $b = 200$	8.395	629,244	5.431

TABLE V. Performance for a buffer size of 200 packets and varying max_p

with the first set of experiments, we observe that Tail Drop increases the dropping probability for the UDP traffic.

C. Fairness

In the following we examined the fairness of RED vs. Tail Drop. Because the traffic consisted of a mixture of different connections with different durations and starting times, we did not calculate a fairness index but instead used the throughput of each connection as a rough measure. Our macroscopic definition of fairness we use here is that all the connections of the same type will get approximately the same throughput.

Figure 2 shows the goodput realized by the different TCP connections we used. We used two different types of FTP/TCP sources, one sending small files, the second sending large files. Since we use only one source type per test all connections should get the same average goodput (see the horizontal line in the plot).

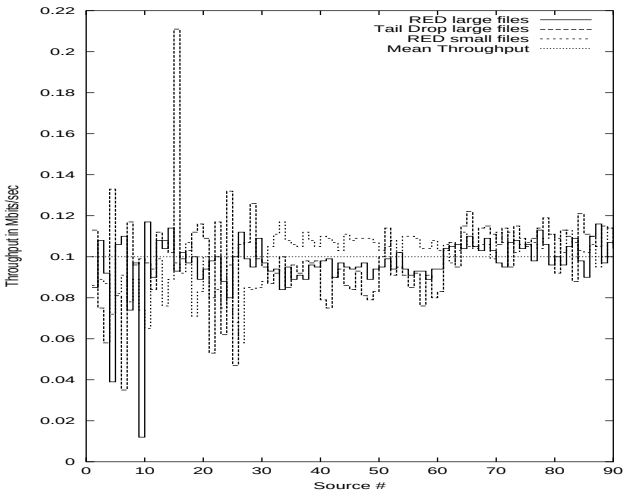

Fig. 2. Throughput for the TCP connections with RED and Tail Drop

These results do not show much difference between the realized throughput with RED and Tail Drop. Neither RED nor Tail Drop results in "fair" throughput for all connections. There are at least two connections getting more than twice (or less than the half) of the mean goodput. This might be due to the fact that RED increases the probability that a packets get dropped. If this occurs during the slow start phase it is very hard to recover from this loss. Therefore, with RED some connections might observe less throughput than others.

D. Increase the buffer size

Next, we examined the RED performance for different buffer sizes. Therefore we compared the TCP and UDP performance for big and small buffers. Unlike in the previous tests, this time, we did not change the values for the minimum ($min_{th} = 10$) and maximum ($max_{th} = 30$) thresholds, but only the total buffer size.

Here we can see a nice RED property. First, the number TCP packets sent is increasing when we increase the buffer size. Second, the bigger the buffer, the fewer drops we can see for the UDP traffic. Third, the drop rate for the UDP traffic is much higher with TD then with RED. This is in contrary to the RED paper, where unresponsive traffic should suffer more with RED then with TD.

E. Impact of the averaging parameter α

Here, we examine how different averaging settings impact the performance of RED. The CISCO routers calculate the average queue size with the following formula:

$$avg_t = avg_{t-1}(1 - \frac{1}{2^\pi}) + queuelength(\frac{1}{2^\pi})$$

The recommended value in [3] is $\alpha = 0.002$. For a CISCO router this corresponds to $\pi = 9$.

In Table VI, we used 5 different settings for the averaging, namely $\pi = 1$ which is the instantaneous queue length, $\pi = 4$, $\pi = 8$ close to the recommended value, $\pi = 12$, and $pi = 16$ no averaging. We used the following RED parameters: minimum Threshold $min_{th} = 10$ and maximum Threshold $max_{th} = 30$ The maximum drop probability was $max_p = 1/10$. The buffer size was set to 100

The authors of RED claimed always that the averaging is a very important feature of the RED algorithm.

Buffer Size	Throughput	Bytes Sent	% UDP drop
$\pi = 1$	8.325	624,737	6.773
$\pi = 4$	8.476	625,738	6.782
$\pi = 8$	8.468	628,941	6.728
$\pi = 12$	8.473	625,742	6.021
$\pi = 16$	8.390	627,520	5.933
TD $b = 100$	8.480	625,828	7.667

TABLE VI. Performance for a buffer size of 200 packets and varying max_p

This smoothing will avoid the bias against bursty traffic. Finally our tests show some opposite results. First, with RED we get more drops for the bursty traffic. Second, the influence of the weighting factor is minimal (or not visible). It seems that the multiplexing of 90 sources smoothes the traffic in a way that the averaging of the queue size has no effect on the TCP nor the UDP performance.

IV. The multiple router case

Setting up the parameters in one single router is already complicated. Tuning RED parameters in a heterogeneous multi-router, multi-ISP environment is even more complex. In this section we want to discuss the problems that might occur when RED will be incrementally deployed in an existing network. Specifically, how RED and non-RED routers perform in concert.

Therefore we added a second router in our local testbed and connected a new Ethernet line to it. The PCs connected to this network together with some of the PC connected to the first router are now used to create cross traffic and increase the load of the two routers (see also Figure 3.

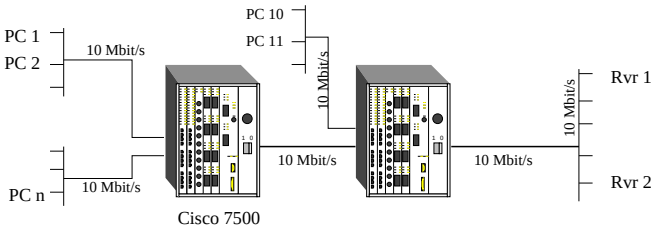

Fig. 3. Testbed Setup for two routers

With this network setup we examined the end2end performance of the flows traversing both routers. We were interested in how the network setup will influence this performance and if an incremental deployment we increase or degrade the measures of interest. In Table VII we used a buffer size of 40 packets with RED and tail drop. In case of RED we used the following setting: $min_{th} = 10$, $max_{th} = 30$, and $max_p = 1/10$.

Setup	Throughput	Bytes Sent	% UDP drop
red-red	8.595	651,635	16.393
red-td	8.558	666,721	25.093
td-red	8.305	646,920	16.538
td-td	8.173	652,204	23.724

TABLE VII. Performance for a buffer size of 40 packets with different network setups

Table VII shows the same measures for a buffer size of 200 packets and the following RED setting: $min_{th} = 30$, $max_{th} = 130$, and $max_p = 1/10$. For both tests we used $\alpha = 0.002$ for the averaging.

Setup	Throughput	Bytes Sent	% UDP drop
red-red	8.511	654,944	14.488
red-td	8.482	654,237	14.915
td-red	8.417	651,038	14.920
td-td	8.648	654,148	14.741

TABLE VIII. Performance for a buffer size of 200 packets with different network setups

This results show an interesting result. While the buffers are small, the incremental deployment is problematic. Changing from a pure tail drop environment to a heterogeneous RED and tail drop network has got an important influence. E.g., UDP traffic losses varying from 16% up to 25%!

The situation is completely different when using larger buffers. Even more, the pure tail drop network performs best compared with the heterogeneous or pure RED networks.

The problem of parameter tuning gets very important when networks like those used in this section are connected without an agreement between the concerned ISPs. Probably, a single ISP or a single misconfigured router can determine the performance for the whole network e.g., by using a setup that degrade the perfomance of the UDP traffic while the other ISP is interested in inceasing UDP throughput.

V. Discussion

The above results prove, at least for the CISCO IOS implementation of RED, that our preferred carrier's reticenses were justified. RED, given the current experimental settings, does not exhibit much better performance than Tail Drop. In the case of UDP traffic, it seems that TD is more aggressive than RED with regard to policing non responsive flows.

We have also shown that the RED parameters have a minor impact on the performance with small buffer. There Using RED with large buffers indeed can improve the systems performance but then choosing good RED

settings is not straight forward. In a multi router case this is differnent. For an incremental deployment, small buffers are very sensitive for the end2end performance.

It might be argued that we merely evaluated a particular RED implementation which perhaps does not fully implement RED. This is, to our understanding, another proof that RED is difficult to calibrate, and that deploying it on a carrier backbone in the current state of understanding would be a mistake.

We believe that, due to the dynamics of the parameters that influence RED, a static RED cannot provide better results than tail drop in the general case.

We therefore recommend the use of more sophisticated active buffer management schemes like the one proposed in [6]. It is shown that a fair queuing environment outperform RED and can give predictable fair bandwidth sharing to all flows.

More complete experiments on larger test environments and heterogeneous RTTs are required. We intend to carry out these experiments in the future. We also intend to complete our evaluation with extensive simulations.

References

[1] Jon Crowcroft and et al. Recommendations on queue management and congestion avoidance. Technical report, End2end Working Group, 1997.

[2] Sally Floyd. Random early detection gateways. http://ftp.ee.lbl.gov/floyd/red.html, August 1993.

[3] Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *Transaction on Networking*, 1993.

[4] Ganymede Software Inc. Chariot 2.2. http://www.ganymedesoftware.com/html/chariot.htm, 1998.

[5] Van Jacobson. Notes on using red for queue management and congestion avoidance. Nanog Workshop, 1998.

[6] Bernhard Suter, T. V. Lakshman, Dimitrios Stiliadis, and Abhijit Choudhury. Efficient active queue management for internet routers. 1997.

[7] CISCO Systems. Ios configuration guide. http://www.cisco.com/, 1998.