Reasons not to deploy RED

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

† INRIA ∗ SPRINT Labs

mmay,bolot@sophia.inria.fr cdiot,lyles@sprintlabs.com

April 8, 1999

Abstract

In this paper we examine the benefits of RED by using a testbed made of two commercially available 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.

1 Introduction

To the question: "would you implement RED on your network?", our preferred carrier answered: "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 (TD)) 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 [5] 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 [3]. 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 [6].

Many of the intuitions which have driven our concern with RED parameter choice 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.

The other observation that has driven our concern about RED is the finding that long TCP transfers are the exception rather than the rule and that most TCP's are in the slow-start phase where packet loss is deadly. Models of TCP performance for short connections such as [1] gave us reason to believe that in a network with a mixture of applications RED might not perform as well as it does in simulations.

2 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 a towards a web server, or real-time audio or 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.

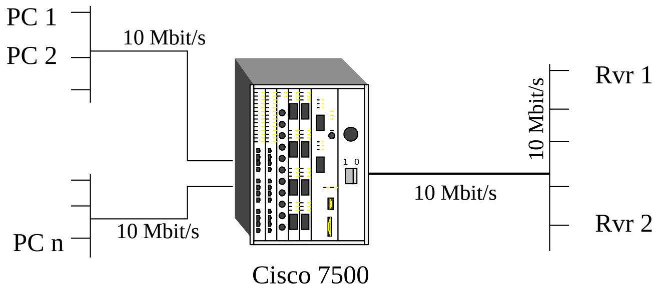

Figure 1: Testbed Setup

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).

3 RED observations 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, [4] 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 [2].

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 *alpha* defines the weight for the past queue size values. ISO allows to set these 4 parameters: the minimum threshold $minth$, the maximum threshold max_{th}, the maximum drop probability max_p, and the averaging parameter $alpha$.

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 ($\%UDPdrop$).

3.1 The maximum drop probability max_p

We start with a *default* RED setting where we chose 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 9, i.e., $alpha = 0.002$.

max_p	Throughput	TCP Bytes	% UDP drop
1/100	8.471	540,705	9.110
1/10	8.304	538,211	9.244
1	8.294	532,909	9.713

Table 1: 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

max_p	Throughput	TCP Bytes	% UDP drop
1/100	8.474	546,722	1.529
1/10	8.337	537,223	1.693
1	8.476	542,620	2.180
TD	8.395	543,619	5.431

Table 2: Performance for a buffer size of 200 packets and varying max_p

small buffer the increase of the drop 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 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.

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

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	TCP Bytes	% UDP drop
5 to 10	8.347	534,805	9.921
5 to 20	8.472	537,706	9.684
5 to 30	8.384	538,908	9.306

Table 3: 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 4. Again, we observe that a variation of the

Interval	Throughput	TCP Bytes	% UDP drop
20 to 50	8.501	547,323	2.000
20 to 100	8.480	545,921	2.013
20 to 150	8.470	547,121	2.685
TD	8.395	543,619	5.431

Table 4: Performance for a buffer size of 200 packets and varying max_p

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

3.3 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

Buffer Size	Throughput	TCP Bytes	% UDP drop
RED 40	8.331	538,010	9.429
RED 100	8.408	543,719	6.407
RED 150	8.579	546,422	4.825
RED 200	8.480	547,423	3.325
TD $b = 200$	8.395	543,619	5.431
TD $b = 50$	8.253	536,689	12.495

Table 5: Performance for a buffer size of 200 packets and varying max_p

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).

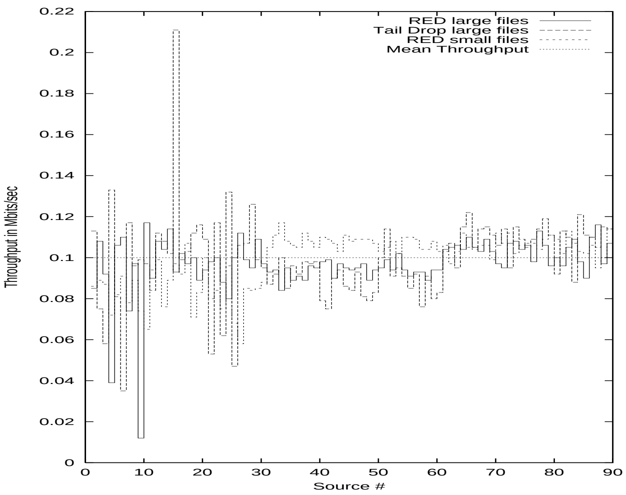

Figure 2: Throughput for the TCP connections with RED and Tail Drop

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.

3.4 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. 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.

4 Discussion

The above results show, at least for the particular implementation of RED studied, that care in making deployment decisions about RED is justified. RED, given the current experimental settings, does not exhibit much better performance than Tail Drop. In the case of our 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. Using RED with large buffers indeed can improve the systems performance but then choosing good RED settings is not straight forward.

It might be argued that we merely evaluated a particular RED implementation which perhaps has a nonstandard implementation of RED. Yet it is the implementations in the marketplace, not the implementations running in simulations, which will determine whether RED has a positive, negative or neutral effect on the Internet. Thus deploying it with 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.

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] Neal Cardwell, Stefan Savage, and Tom Anderson. Modeling the performance of short tcp connections. Technical report, 1998.

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

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

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

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

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

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