Aggregate Traffic Performance with Active Queue Management and Drop from Tail

Gianluca Iannaccone
Sprint ATL
1 Adrian Court
Burlingame CA 94010
gianluca@sprintlabs.com

Martin May
Activia Networks
Parc of Sophia Antipolis
06225 Vallauris Cedex, France
martin.may@activia.net

Christophe Diot
Sprint ATL
1 Adrian Court
Burlingame CA 94010
cdiot@sprintlabs.com

ABSTRACT

Active queue management (AQM) refers to a family of packet dropping mechanisms for router queues that has been proposed to support end-to-end congestion control mechanisms in the Internet. In this paper, we examine the performance of AQM mechanisms by varying two parameters: the queue size and the dropping function. AQM flavors considered include "RED", the more recently proposed "Gentle RED" and an additional mechanism we call "Gentle RED with instantaneous queue size".

We use experimentation to analyze the performance of the AQM mechanisms identified above on the aggregate traffic going through a congested router. The metrics used are: TCP goodput, TCP and UDP loss rate, queueing delay and consecutive loss probability. The AQM mechanisms are compared to Drop from Tail, the buffer management mechanism currently found in most operational routers.

The major observation is that AQM mechanisms have a minor impact on the aggregate performance metrics we observe. On the other hand, we observe an important sensitivity of the AQMs considered to traffic characteristics that may compromise their operational deployment.

1. INTRODUCTION

Active Queue Management (AQM) refers to a family of packet dropping mechanisms for router queues. AQM has been designed to support end-to-end congestion control in packet networks. The principle of AQM is to pro-actively drop packets in a router in anticipation of congestion. Such packet losses are further interpreted (through acknowledgements or timeouts) by TCP sources as a request to reduce their sending rates [18].

The AQM mechanisms that we are studying are purely FIFO based. Packets are dropped based on the load of the flow aggregate, with no knowledge of the individual flows that compose that aggregate.

Per-flow queueing is another family of control mechanisms which provides feedback based on individual flow status [6]. Per-flow queueing is significantly more complex than FIFO based AQM mechanisms and is not currently deployed in operational IP networks. Until RED was proposed by the "end-to-end" IRTF group [2], Drop from Tail (or DT) was the only mechanism used in network nodes. Drop from Tail remains the most popular mechanism in IP routers today, mostly because it is robust, and because it is simple to implement. In DT, all packets are accepted in the FIFO queue until it is full. Packets arriving to a full queue are dropped. DT has a number of drawbacks. It may for example result in an unfair treatment of individual flows, and may also cause flow synchronization.

RED has been designed to eliminate some of the DT problems. RED was initially described and analyzed in [12]. It enhances Drop from Tail by introducing two new elements in the queue management scheme: (i) the average queue size estimation, and (ii) a dropping function.

The *average queue size* is used to define the probability with which packets are dropped (instead of the instantaneous queue size as in Drop from Tail). The average queue size is estimated using an exponential weighted moving average:

$$\hat{k} \leftarrow (1-w)\hat{k} + w\ k,$$

where w is a fixed parameter with a small value ($0 \leq w \leq 1$) and k is the instantaneous queue size. The value of w determines the responsiveness of the system to variations in the input traffic; the average queue size is therefore intended to smooth router's reaction to traffic fluctuations. Note that the calculation of the average queue size adds complexity to the buffer management algorithm, since the new average has to be calculated periodically.

The *dropping function* is used to anticipate congestion by dropping packets before the FIFO queue is full, instead of waiting for the buffer to overflow. The parameters of the dropping function were first defined in [12]. A first adjustment of the RED parameters was proposed in [13]. The second modification [14] introduced a new parameter in the dropping function.

Figure 1 illustrates the evolution of the RED dropping function. The value min_{th} specifies the average queue size below which no packets are dropped, while max_{th} specifies the average queue size above which all packets are dropped. As the average queue size varies from min_{th} and max_{th}, packets are dropped with a probability that varies linearly from 0 to max_p. The uppermost graph in figure 1 corresponds to the dropping function used in [12]. After further simulations, it was considered necessary to increase the maximum drop probability from $max_p = 0.02$ to $max_p = 0.1$ (2% drop probability was not enough to force multiple TCP sources to sufficiently reduce their window sizes) [13]. Based on findings in [24], a modification has been lately introduced in the dropping function to make the drop probability increase more "smoothly" between max_p and 1 [14] when the average queue size is above max_{th}.

In this paper we refer to the AQM mechanism described in [12] and further modified in [13] as "RED"; "Gentle RED" refers to the mechanism described in [14].

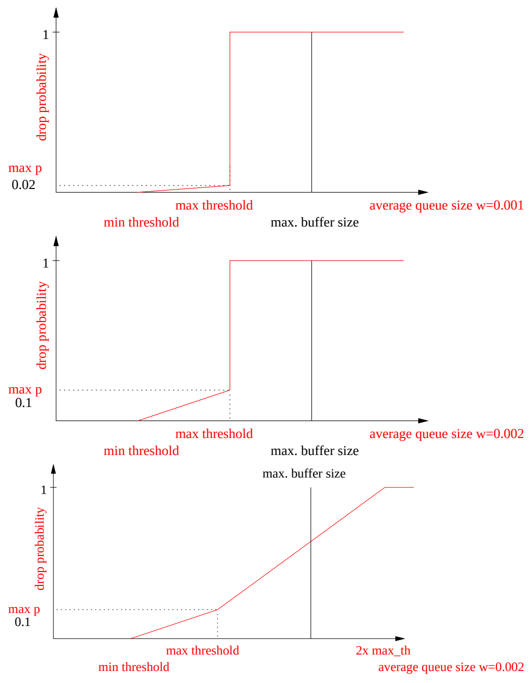

Figure 1: The evolution of the RED dropping functions.

We characterize AQM mechanisms using the two parameters introduced above: the dropping function and the queue size. We propose an analysis of the AQM design space based on these two parameters. Table 1 represents the AQM design space we have defined. In this context, we analyze RED, "Gentle RED" (or GRED) as well as an additional mechanism we call "Gentle RED with instantaneous queue size", or GRED-I. Drop from Tail is also part of our framework as a buffer management mechanism characterized by a sharp dropping function based on the instantaneous queue size.

	avg. queue size	inst. queue size
sharp dropping function.	RED	Drop from Tail
smooth dropping function.	Gentle RED	Gentle RED instantaneous

Table 1: Different buffer management algorithms.

We focus our interest on aggregate traffic crossing a router, instead of analyzing the performance of individual TCP flows, for the following reasons:

- All experimental results we obtained on individual TCP flows are not different from previous results [5, 19, 20].
- The aggregate traffic behavior in a router represents an important metric for Internet Service Providers (ISP). ISPs are interested in maximizing the goodput and minimizing the delay of the traffic traversing their networks. There is no such evaluation available for RED in the literature.

Note that we do not consider Active Queue Management schemes for traffic differentiation such as RIO or WRED. In these schemes, the dropping function plays a completely different role than in RED; it differentiates between traffic classes rather than being primarily a congestion control mechanism.

The rest of the paper is organized as follows. Section 2 summarizes related literature. In Section 3, we define the metrics we have chosen to analyze AQM mechanisms. Then we describe our evaluation environment which is based on an experimental testbed. In Section 4 we compare DT, RED, GRED, and GRED-I using the parameter values proposed in [13] and [14]. We show that AQM performances are very similar for the metric considered. On the other hand, we demonstrate and examine the sensitivity of AQM to the nature of the traffic (number of flows, and whether it is made of long-lived or short-lived TCP connections). We conclude this paper with a summary of findings and discussion of future work.

2. RELATED WORK

Despite the popularity of RED, very few detailed studies of its performance have been published. We are not aware of any operational large scale IP network relying on RED. One of the few published measurement studies [7] is limited in scope as it only considers the router performance (as opposed to the end-to-end performance) and it does not clearly describe the measurement settings and the exact information being measured. In most publications, the authors use simulations to evaluate RED and examine the impact of parameter choice on end-to-end performance [4, 8, 9, 10, 20]. But, realistic simulation settings are hard to come up with: they require choosing many parameters (network topology, bandwidth, traffic matrix, traffic source types, etc.), and, even taking into account recent advances in traffic analysis and generation.

Extensions or changes to the RED algorithms, such as FRED [19] or SRED [21], have been proposed to make it more robust or adaptive. However these papers do not question or evaluate the performance and suitability of the RED algorithm. Consequently, it is not yet clear how to choose AQM parameters. Recommendation for choosing parameter values can be found in [13] and [14]. These values also rely on simple experiments as well as on the authors' intuition.

More recently, in [11], the authors use an analytical approach to choose appropriate RED parameters. The analytical approach of this paper, while providing useful insight into the impact of RED parameters, may not be able to capture the many interactions that arise in real networks and realistic experimental settings such as ours. In [17], a control theoretic approach is used to study RED stability and tradeoffs involved in various parameter choices, while in [25], the authors derive a quantitative model for setting RED parameters in presence of TCP traffic. Both models provide basic guidelines to set RED parameters in order to guarantee few oscillations of the queue size depending on the traffic characteristics (namely, round trip time and number of flows).

In [5], Christiansen et al. propose an experimental analysis of RED when used with web traffic. This study, done with a large number of short-lived TCP flows questions the performance of RED established earlier by simulation. Our study complements [5] by providing experimental performance analysis with different types of traffic, and by making observations that are applicable to the entire AQM space, and not just RED.

3. EVALUATION ENVIRONMENT

Our analysis of AQM is primarily based on experimental results using routing equipment currently deployed and widely used in operational networks.

3.1 Metrics

In this paper, we do not consider per flow performance or fairness amongst TCP flows, although these measures might well be important for service level agreements or customer satisfaction. Instead, we are interested in how *the aggregate of multiple flows* is affected by an AQM scheme. Consequently, we define specific metrics to accurately describe the effects of different AQM schemes in a router on traffic aggregate:

- *TCP goodput at a router interface*. This metric reflects the best use of the available router resources. We define TCP goodput of the traffic aggregate as the bandwidth delivered to all TCP receivers, excluding duplicate packets. We are interested in verifying if the use of Drop from Tail or other AQMs may affect TCP goodput due to suboptimal link utilization.

- *TCP and UDP loss rate*. Loss rate is defined as the number of dropped packets divided by the total number of packets arrived at the router's input ports. While TCP goodput and loss rate are somewhat redundant metrics, loss rate covers a very important role for applications using UDP. We choose to differentiate between TCP and UDP loss rates to verify AQM mechanisms' behavior against different kind of traffic sources.

- *Queueing delay*. We examine the variation of the router output queue size over time, in order to measure the average queueing delay and its standard deviation. Minimizing average queueing delay is an important goal for real-time applications. On the other hand, the standard deviation gives a measure of the jitter at the destination hosts. Jitter degrades performance for (i) TCP flows, as TCP calculates the variance to determine the spacing of the TCP ACKS and (ii) audio/video flows, as those applications have to use a large playout buffer at the receiver to compensate for high jitter.

- *The number of consecutive losses*. The interaction of TCP congestion control mechanism with many small file transfers results in bursty traffic. Loss patterns with DT follow the burstiness of input traffic, as all incoming packets are dropped until the packet at the head of the queue is served. On the other hand, RED is expected to spread out packet drops, since it randomly drops packets to anticipate congestion. This metric measures the probability of having consecutive packet drops at a single router interface.

3.2 Experimental environment

The testbed topology (Figure 2) represents a gateway where many networks are merged into one outgoing link, hence creating a potential congestion point. Routers used on the testbed are CISCO 7500 routers running IOS 12.0. In this configuration the bottleneck link is between the two routers. At the egress router, no packet will be dropped as the maximum arrival rate is 10MBit/s and the links between the second router and the traffic sinks have sufficient capacity to handle the arriving traffic.

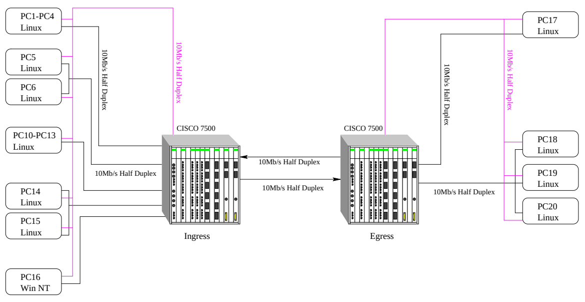

Figure 2: Testbed setup for measurements with two routers.

This is a common configuration on edge routers, where ISPs can practice over-subscription: the cumulative bandwidth of customer links entering an edge router is greater than the bandwidth of the link going to the backbone router. This is typical in xDSL networks, for example. Over-subscription relies on the observation that all customers do not use 100% of their bandwidth at the same time. If, for some reason, the total bandwidth of the ingress links is higher than the bandwidth of the egress link, congestion occurs. On the other hand, backbone links are over-provisioned and congestion does not happen on the backbone.

Thus, we are basically studying the performance of AQM at a congested gateway between a backbone and an enterprise network.

Although the type of router we use in this testbed is among the most commonly used ones in the Internet, there are two major drawbacks in using commercial equipment. First, results obtained from the testbed are dependent on the vendor implementation of RED. This makes it difficult to generalize our observations. Second, it limits the kind of metric we can study to those provided by the router vendor. Therefore, we have designed a second testbed where AQM mechanisms are implemented in the Dummynet link emulator [23] (Figure 3). In this testbed, the point of congestion is not the first router, but the Dummynet box located between the two routers. AQMs implementations in Dummynet follow the exact specifications found in [13] and [14]. Dummynet is also a flexible environment where it is possible to study a wider variety of metrics than in a commercial router (e.g. queueing delay and consecutive loss distribution).

Traffic is injected by high-end PCs running Linux (kernel version 2.2.x). Linux implements the TCP SACK congestion control mechanism. In both testbeds, data sources are the PCs to the left side, while data sinks are the hosts on the right side. Network links are 10Mbps Ethernet links. Between the two routers, we have two half-duplex Ethernet links to ensure that Ethernet collisions do not degrade performance. In the router testbed, these Ethernet links are 10Mbps. In the Dummynet testbed, the link going from the ingress router to the Dummynet box is a 100 Mbps Ethernet. The Dum-

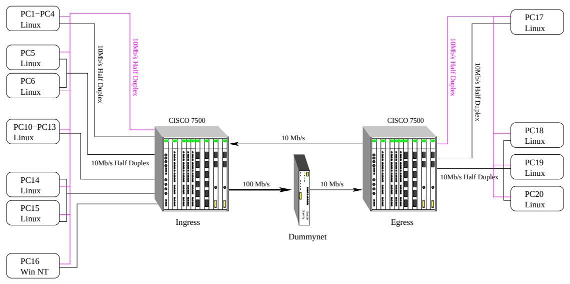

Figure 3: Testbed setup with Dummynet.

mynet box emulates thus a 10 Mbps bottleneck link. Other links are the same for both testbeds.

We examine the AQM schemes and DT in the ingress router and Dummynet, respectively. The AQM mechanism in the egress router is Drop from Tail with a very large FIFO buffer.

We also introduced additional propagation delay on some of the Ethernet links between the sources and the first router. Hence, different connections experience different transmission delays, varying between 120ms and 200ms[1].

For all experiments, we used a router buffer size of 200 packets. This corresponds approximately to the bandwidth-delay product of our testbed[2]. For all experiments with RED we used the parameter values recommended in the CISCO IOS online documentation [3] which are the same as those proposed in [13]:

- minimum threshold $min_{th} = 30$,
- maximum threshold $max_{th} = 130$, and
- maximum drop probability $max_p = 0.1$.

It is straightforward to verify that the set of RED parameters shown above, given the network traffic described in Section 3.3, also follows the guidelines proposed by Hollot et al. [17] (see Appendix A).

For GRED and GRED-I, we use parameters defined in [14]. Since GRED is not yet implemented in commercial routers, we used an approximation of GRED to evaluate its performance on the router testbed. Figure 4 shows the shape of the drop function for GRED and our approximation. On the Dummynet testbed, we use an exact implementation of GRED and GRED-I.

Note that our goal is to study aggregate traffic performance of different AQMs from the point of view of an Internet Service Provides (ISP). Therefore, we do not use different RED parameters for different traffic scenarios, as it would be unrealistic in the case of the deployment of AQM in an operational network.

3.3 The traffic

[1]Note that we could only vary the delays of a complete link, not of individual flows. Therefore all flows using a specific link suffer the same artificial propagation delay.
[2]TCP packets are 1500 bytes long, while UDP packets are 573 bytes long.

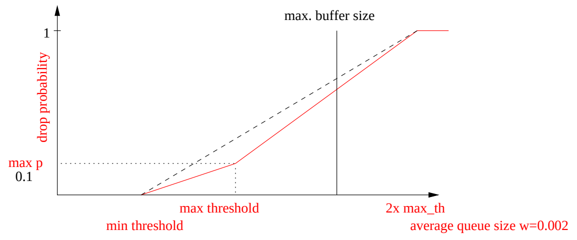

Figure 4: The shape of the GRED drop function (plain lines) and our approximation (dashed lines) on the router testbed.

We use the Chariot 3.2 [16] tool to generate the traffic. Chariot is an application emulator that can generate various applications' traffic (HTTP, FTP, audio streaming, etc.) using the host protocol stack implementation. Each individual flow can have specific parameters (file size, packet size, starting and ending time) in order to compose complex traffic sources.

In this paper we analyze two types of network traffic:

- A traffic mix made of a set of long-lived TCP connections with a fixed amount of UDP traffic (1Mbps, 10% of the bottleneck bandwidth). A long-lived TCP connection is a large FTP transfer that starts at the beginning of the experiment and persists for the entire duration of the experiment. We run a set of experiments varying the number of TCP connections (from 16 to 256 active connections), to study the behavior of the AQMs and DT for different average offered loads. Although this type of traffic is very unrealistic, it makes it easy to vary the number of TCP connections in the traffic aggregate and to compare our results to simulation studies found in literature.

- A more realistic traffic mix made of many short-lived TCP flows (to model, for example, HTTP responses), and a few long-lived ones (e.g. large file transfers, MP3 files). Non-responsive UDP traffic (e.g. audio/video applications) still represent 10% of the bottleneck link bandwidth. In these experiments, the number of active TCP connections is 256, with 32 connections being long-lived and the remaining being short-lived. Given that the number of active TCP connections is fixed, a short-lived connection that ends is immediately replaced by a new one between the same pair of end-systems. Short-lived TCP connections transfer 20Kb files. Note that the number of concurrent flows is consistent (appropriately scaled down) with the number of flows observed on an OC-3 link on the Sprint IP backbone [15]. Moreover, the ratio between long-lived and short-lived flows is also consistent with the figures given in [15] where approximately 90% of the flows are made of less than 20 packets.

The main benefit of using an experimental testbed instead of a simulated environment is that we can evaluate the impact of AQMs using real equipment and actual implementations of the algorithms, avoiding any assumptions or simplifications on network behavior. In addition, we can avoid undesirable phenomena typical of simulations, such as synchronization and phase effects, due, for example, to inaccuracies in service time models, or to the absence of asynchronous events in the end-systems and routers.

4. EVALUATION OF AQM MECHANISMS

In this section, we study the performance of DT, RED, GRED, and GRED-I. As shown in Table 1, these four mechanisms cover the spectrum defined by the two parameters we have chosen to describe AQMs. GRED-I (Gentle RED with instantaneous queue size) is a mechanism that combines the dropping curve of GRED with an instantaneous measure of the queue size such as in DT.

We run each experiment for 10 minutes and repeat each experiment at least 20 times (10 with the router testbed, and 10 with the Dummynet testbed). All TCP connections are started at random times uniformly distributed in the first 10 seconds. Then, short-lived TCP connections are restarted as soon as they finish, while long-lived ones last for the entire duration of the experiment.

The difference between the two testbeds in the metrics observed is always minimal and contained within the 95% confidence intervals. Unless explicitly specified, all graphs in the following sections show the average of 10 runs on the Dummynet testbed.

4.1 Goodput and packet loss rates

Figure 5 plots the goodput of all TCP connections with the congestion point (i.e. the ingress router or the Dummynet box) running the three AQM mechanisms and DT by turn. Figure 6 shows the average loss rates at the bottleneck link for the same set of experiments[3]. The traffic is made of 16 to 256 long-lived TCP connections with 1Mbps of UDP traffic.

As we can see from the graphs, it is difficult to differentiate AQM mechanisms based on their aggregate goodput or packet loss rates. Goodput is comparable for AQMs and DT in all traffic scenarios, except the case with only 16 TCP connections. Indeed, in presence of few connections, DT provides a higher goodput that the other three AQMs. This relatively small difference (less than 5% for 16 TCP connections and under 1% otherwise) is imputable to the high value of max_p that makes the few TCP sources back off too often despite the low degree of congestion that the network is actually experiencing. We ran a few experiments with the max_p parameter equal to 0.01 and all the AQMs showed the same performance of DT with 16 TCP connections.

Figure 6 illustrates that AQMs' goal of keeping buffer occupancy low comes at the cost of a higher overall packet loss rate. Note however that higher loss rates do not negatively impact goodput thanks to the shorter queueing delay experienced by TCP connections. The minimum loss rate is observed most of the time for DT, which is consistent with our intuition. Also note that because the offered load increases with the number of TCP connections, the loss rate increases accordingly.

In summary, we make the following observations from the analysis of the TCP goodput:

- All the mechanisms considered provide almost identical performance.
- Parameters in RED, GRED, and GRED-I can be tuned for each new traffic condition to perform like (or slightly better than) DT.

[3]The graphs are the result of averaging 10 experiments and the 95% confidence intervals are shown.

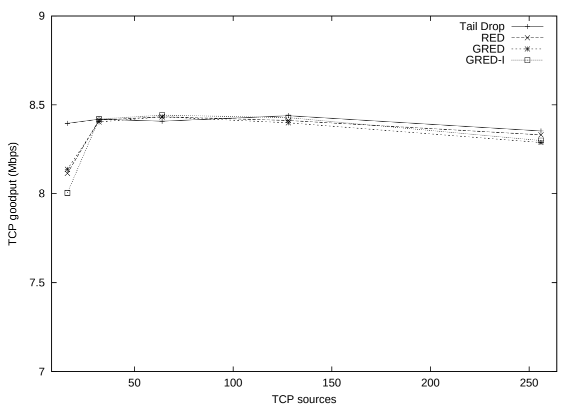

Figure 5: Average TCP goodput with DT, RED, GRED and GRED-I for long-lived TCP connections.

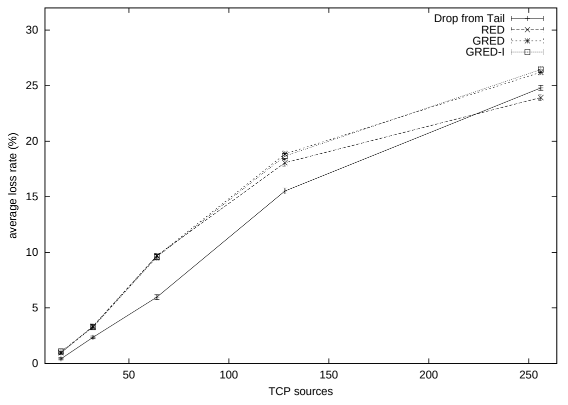

Figure 6: Average loss rate with DT, RED, GRED and GRED-I for long-lived TCP connections.

- DT exhibits lower loss rates compared to AQMs. However, higher loss rates in AQM do not impact TCP goodput, given the shorter queueing delay experienced by packets.
- Link utilization is always very high. Even with very few TCP flows we do not see a lower link utilization with DT, as described in previous works [2, 12].

In Figure 7, we observe TCP goodput and TCP/UDP loss rate for 256 long-lived TCP connections with 1Mbps of UDP traffic. In Figure 8, we plot the same metrics for the more realistic traffic mix of short and long-lived TCP connections.

Figures 7 and 8 confirm our previous observations about TCP goodput. The difference in TCP and UDP loss rate is always very small. For all mechanisms the loss rate is much higher for a traffic made of long-lived flows only (between 25% and 30%) than for a traffic made of short and long-lived flows (between 13% and 17%). This result is expected since short-lived TCP connections offer a lower load to the network; indeed, a short-lived connection will spend most of its lifetime in the slow-start phase with a relatively small congestion window.

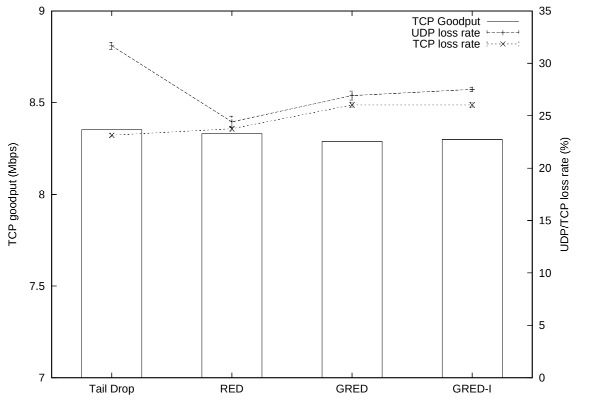

Figure 7: TCP goodput (left) and TCP/UDP loss rate (right) for 256 long-lived TCP connections with 10% of UDP traffic.

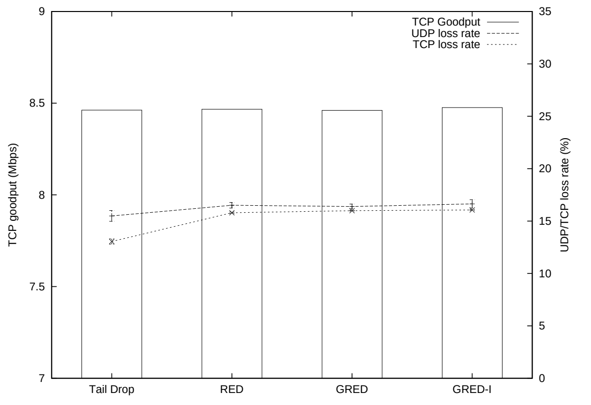

Figure 8: TCP goodput (left) and TCP/UDP loss rate (right) for a mix of 256 short and long-lived TCP connections with 10% of UDP traffic.

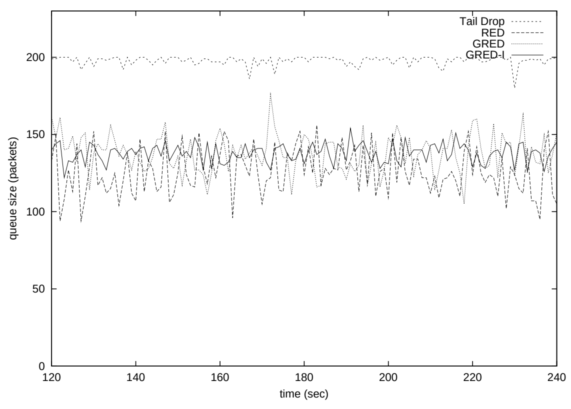

Figure 9: Queue size vs. time with DT, RED, GRED and GRED-I, for 256 long-lived TCP connections.

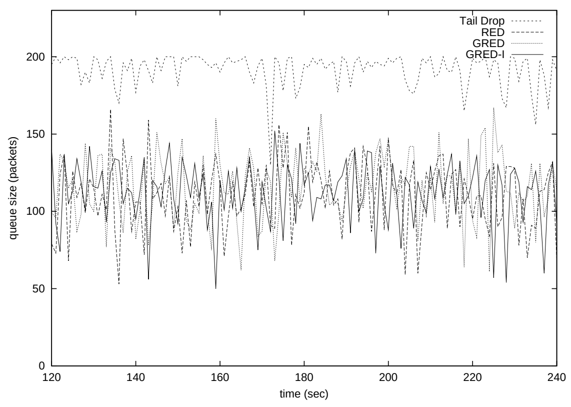

Figure 10: Queue size vs. time with DT, RED, GRED and GRED-I, for 256 short and long-lived TCP connections.

The difference in loss rate between TCP and UDP with 256 long-lived flows and Drop from Tail is counter-intuitive. We have identified this behavior as being a consequence of the size of UDP packets used for our experiments (i.e. 573 bytes). Since the queue size (counted in packets) is always very close to the maximum buffer size (Figure 9), flows using many small packets will likely suffer more packet losses than flows using few large packets. When the UDP packet size is 1500 bytes, we observed the same loss rate for TCP and UDP (note that TCP packet size is also 1500 bytes).

4.2 Queueing behavior

In this section, we study experimentally the queueing behavior. Figure 9 and Figure 10 plot the evolution of the queue size with DT, RED, GRED, and GRED-I for the a traffic made of long-lived connections and the mix of short and long-lived connections, respectively[4].

To analyze the queueing behavior, we have computed the average delay and its standard deviation for all mechanisms (Tables 2 and 3). Results are averaged over 10 experiments.

[4]The graphs are the result of sampling the queue size every second for a 2 minutes period after the network has reached a “steady” state.

The standard deviation is less than 25% of the average delay in both traffic scenarios, for all AQMs and DT. In particular, for DT and GRED-I, the standard deviation is around 8ms for long-lived connections; it is twice as much for RED and GRED. For short-lived connections, the standard deviation is higher and almost equal for all mechanisms (around 21ms for DT, RED and GRED, and 17ms for GRED-I). We conjecture that the difference in the standard deviation between the two traffic mixes is due to short-lived TCP connections that are too short to enter congestion avoidance, and thus inject a burstier traffic in the network.

However, we cannot observe global synchronization phenomena for DT, as described in [13]. Higher oscillations are observed for RED and GRED than for DT and GRED-I. We conjecture that this is due to the use of the instantaneous queue size instead of the aver-

	DT	RED	GRED	GRED-I
average delay	209.55	130.71	144.60	144.27
standard deviation	6.94	16.91	16.06	8.42

Table 2: Average delay and standard deviation (in ms) for 256 long-lived TCP connections.

	DT	RED	GRED	GRED-I
average delay	168.59	94.92	100.56	99.87
standard deviation	21.18	22.10	20.62	16.73

Table 3: Average delay and standard deviation (in ms) for 256 short and long-lived TCP connections.

age queue size in the computation of the dropping probability. Indeed, for DT the queue is always almost full, making the queueing delay less variable, while with GRED-I the use of the instantaneous queue size makes the algorithm to respond much faster to variation in the buffer occupancy.

In terms of average delay, DT converges towards the queue size and RED, GRED, and GRED-I toward the value of max_{th}. This result is predictable as the first property of an AQM mechanism is to define an upper bound to the size of the queue, i.e. max_{th}. Changing the number of buffers to max_{th} in DT would make DT perform like RED in terms of average queueing delay.

In summary, AQM mechanisms act as delay limiters. Modifying AQM parameters mostly influences the average transmission delay through a router, and a RED router behaves essentially like a Drop from Tail router with buffer size max_{th} [1]. Finally, we have observed that the use of the instantaneous queue size in the computation of the dropping rate seems to reduce the standard deviation of the queueing delay.

4.3 Consecutive losses

To investigate further the loss distribution, we study consecutive losses. Experiments are performed on the Dummynet testbed only because consecutive losses are not available on Cisco routers. Note that consecutive losses do not clearly indicate that a mechanism is performing poorly. But in the common understanding, long bursts of losses may result in higher jitter and in unfairness among flows.

We define the *length* of a loss sequence as the number of packets N being dropped before a new packet is accepted and enqueued. In Figure 11 we plot the probability of having a loss sequence length greater than n for a traffic made of 256 long-lived TCP connections.

The probability of suffering long sequences of losses appears to be higher with RED than with DT. The probability of losing more than 6 packets in a row, for example, is one order of magnitude higher with RED than with DT. RED also exhibits much longer loss sequences than any of the 3 other mechanisms[5].

This result is counter-intuitive and can be explained as follows. In DT, long bursts of packets cannot be lost as a new packet is accepted in the queue as soon as a packet has been served. Thus, the maximum duration of a loss event is a function of the arrival rate and the output rate. In RED, instead, packet losses are defined by a dropping function. As the average queue size reaches max_{th}, the dropping function tells RED to drop packets, whether room is available or not in the queue. In Figure 12 we plot the average queue size and instantaneous queue size over time in one of the

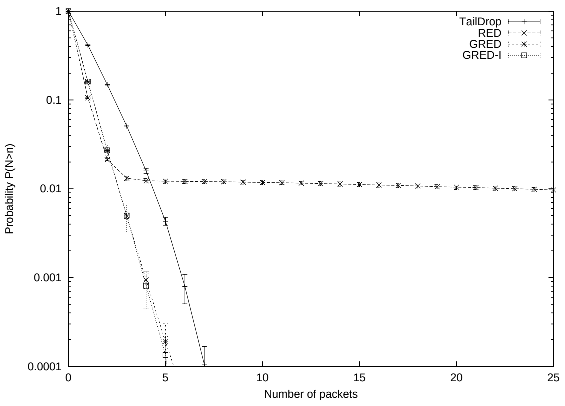

Figure 11: Probability of having a loss sequence length greater than n with Drop from Tail, RED, GRED, and GRED-I, with 256 long-lived TCP connections (95% confidence intervals are shown).

experiments with 256 long-lived TCP connections where RED is the buffer management algorithm[6]. As we can see from the graph the average queue size is always very close to max_{th}. Thus, in the event of a sequence of packet bursts, RED will likely accommodate the first burst, but given that the average queue size will reach max_{th}, it will drop any subsequent packet. A packet will be enqueued only after the average queue size, controlled by the first-order low-pass filter, goes below max_{th} (i.e. after the first burst of packets has been served and the *actual* queue size has moved below max_{th}).

Note that both GRED and GRED-I exhibit better performance than DT and RED. In this case, the use of a smoother dropping function allows to spread out losses, since the probability to drop a packet is always less than 1.

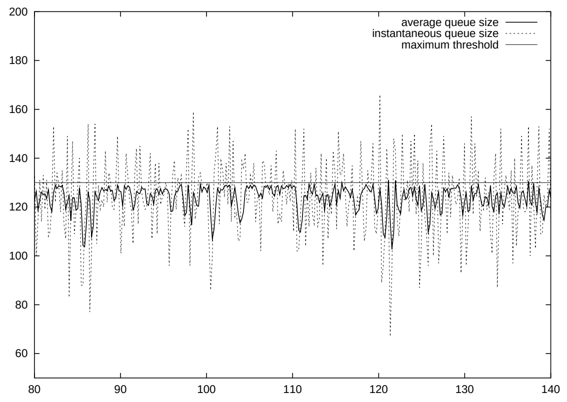

Figure 12: Average queue size and instantaneous queue with RED for a traffic made of 256 long-lived TCP connections.

Choosing appropriate RED parameters would reduce the difference between RED and DT. Using "optimal" parameter values as defined in [25], we can reduce the probability of having long loss sequences. REDopt (Figure 13) uses the following parameters: $min_{th} = 50$, $max_{th} = 200$, $max_p = 0.23$, $w_q = 0.01826$ and a

[5]Note that to ease reading of graphs we only plotted loss sequences with a length shorter than 25 packets, even though in our experiments RED has dropped up to 93 packets consecutively.

[6]The graph is the result of sampling the queue size and the average queue size every 500ms.

buffer size of 300 packets. In order to allow a meaningful comparison with DT and RED, we scaled up their parameters accordingly (i.e. buffer size of 300 packets, $min_{th} = 45$, $max_{th} = 195$).

RED now performs better than DT for short sequences of losses, but very long sequences remain more probable with RED. Note that changing only the buffer size (and the min_{th}, max_{th} range) does not influence RED or DT performance.

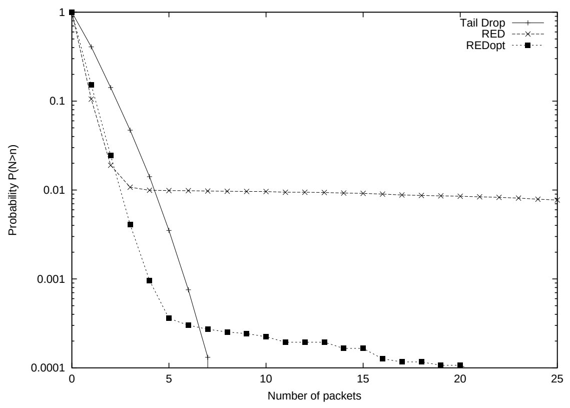

Figure 13: Probability of having a loss sequence length greater than n with DT and RED with 256 long-lived TCP connections and with "standard" and "optimal" RED parameters.

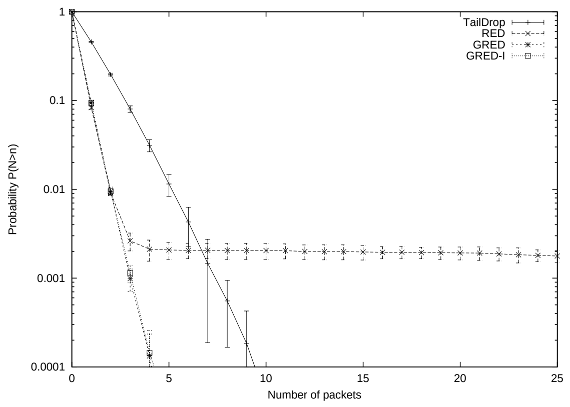

Figure 14: Probability of having a loss sequence length greater than n with Drop from Tail, RED, GRED, and GRED-I, with a mix of short and long-lived TCP connections (95% confidence intervals are shown).

Figure 14 shows AQMs' performance in terms of consecutive losses for a traffic mix of 256 short and long-lived TCP connections. We conjecture that the presence of more bursty sources (short-lived TCP connections during the slow-start phase) increases the probability of long loss sequences for DT, while it has the opposite effect for RED, GRED and GRED-I. As we said above, DT loss patterns depend on the ratio between the arrival rate and the output rate. Therefore, the presence of bursty traffic may be a reason of the increase in the average length of loss sequences.

On the other hand, for RED, GRED and GRED-I, the average queue size is always well below max_{th}, thus reducing the likelihood of long bursts of losses. However, in terms of the maximum length of loss sequences, RED remains the worst option.

As a final note, GRED and GRED-I perform consistently better than DT and RED with both long-lived and short-lived TCP connections, suggesting that the use of a smooth dropping function reduces the occurrence of consecutive packets losses.

5. CONCLUSION

In this paper we have examined the impact of Active Queue Management schemes on the aggregate performance of multiple TCP and UDP flows crossing a congested gateway. Our testbed is representative of the complexity of a gateway between a tier-1 IP backbone and a set of corporate customers. Aggregate performance through a router is significant to an ISP whose network attempts to carry the maximum amount of data with the shortest delay.

Our analysis of the AQM design space is based on two parameters: the dropping function and the queue size. Three AQM mechanisms have been compared to Drop from Tail. We have run experiments using two independent implementations of RED, one (proprietary) available in Cisco's routers and the other (public) implemented over the Dummynet tool following the specification described in [13] and [14]. Both test environments provide similar results.

We can summarize our observations as follows:

- The analysis of the aggregate TCP goodput does not allow to differentiate AQMs from DT. In particular we have not been able to replicate global synchronization phenomena described in previous works [2, 12], that should be responsible for underutilization of network resources.
- A small difference on RED parameters can have a large impact on aggregate performance. This result confirms that choosing RED parameters might be a real challenge in an operational router where the traffic fluctuates due to time-of-day effects. A single set of RED parameters may perform very well in certain traffic conditions and be harmful as the traffic changes in number of flows or offered load. It should also be noticed that the set of RED parameters that optimizes the TCP goodput for a given traffic mix is not necessarily the same set of parameters that optimizes RED for consecutive losses for example.
- AQM mechanisms provide a shorter average queueing delay at the cost of a higher loss rate compared to DT. However, TCP goodput is not affected. Basically, a RED router behaves like a Drop from Tail router with a buffer size set to max_{th} [1].
- The analysis of standard deviation of queuing delay shows that there is a clear advantage in computing packet loss on the instantaneous queue size. On the other hand, observations on consecutive losses shows a significant advantage to mechanisms implementing a smooth dropping function. Thus, RED generally exhibits the worst performance and GRED-I, our newly proposed mechanism, generally exhibits better performance than RED, GRED, and DT.

Thus, in our opinion, Drop from Tail remains the most appropriate mechanism to control traffic on current IP networks. Drop from Tail is easy to implement (no parameter, no computation costs) and easy to predict (the buffer size being known). We did not find any incentive to deploy RED in this analysis. GRED-I would be a viable alternative option as it performs consistently better than DT and does not seem to be as sensitive as RED to changes in traffic characteristics.

Even in the context of the Explicit Congestion Notification (ECN) proposal [22], marking packets based on Drop from Tail (with a marking threshold lower than the queue size) might be sufficient to signal congestion in the network. We suspect that the shape of the dropping function will be even less significant with ECN, as the congestion feedback is computed by the receiver based on the ECN marks, and forwarded as an explicit signal to the sender. Thus, a receiver can apply a feedback policy that takes into consideration the ECN marking function bias, allowing for significant simplification at the router level (we make the assumption that all routers implement the same buffer management scheme).

In addition, we do not believe that the solution to a more efficient use of the network resource is in the AQM space. Drop from Tail will remain the simplest and the most appropriate buffer management mechanism as long as the backbone network is overprovisioned. On access networks, more sophisticated techniques are instead required. Thus, we will investigate how simple Fair Queueing schemes perform and compare these observation to AQM evaluation. Early observations on the number of flows on Sprint IP network seem to make Fair Queueing an attractive solution [15].

6. REFERENCES

[1] T. Bonald, M. May and J. Bolot, Analytic Evaluation of RED Performance, in *Proceedings of IEEE Infocom*, 2000.

[2] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, L. Zhang, Recommendations on Queue Management and Congestion Avoidance in the Internet in the Internet, RFC 2309, April 1998.

[3] CISCO-Systems, IOS Configuration Guide, 1998 `http://www.cisco.com`.

[4] D. Clark, Explicit Allocation of Best Effort Packet Delivery Service, Technical report, MIT Laboratoty for Computer Science, 1997.

[5] M. Christiansen, K. Jeffay, D. Ott and F. Donelson Smith, Tuning RED for Web Traffic, in *Proceedings of ACM Sigcomm*, 2000.

[6] A. Demers, S. Keshav and S. Shenker, Analysis and simulation of a fair queueing algorithm, SIGCOMM Symposium on Communications Architectures and Protocols , October 1989.

[7] S. Doran, Interface Graphs of a RED-enabled router, `http://adm.ebone.net/~smd/red-1.html`, 1998.

[8] W. chang Feng, D. D. Kandlur, D. Saha, and K. G. Shin, Understanding TCP Dynamics in an Integrated Services Internet, in *Proceedings of NOSSDAV*, 1997.

[9] W. chang Feng, The Impact of Active Queue Management on Multimedia Congestion Control, in *Proceedings of IC3N*, 1998.

[10] W. chang Feng, D. D. Kandlur, D. Saha, and K. G. Shin, BLUE: A New Class of Active Queue Management Algorithms, Technical report, Department of EECS Network Systems Department University of Michigan, 1999.

[11] V. Firoiu and M. Borde, A Study of Active Queue Management for Congestion Control, in *Proceedings of IEEE Infocom*, 2000.

[12] S. Floyd and V. Jacobson, Random Early Detection Gateways for Congestion Avoidance, *IEEE/ACM Transaction on Networking* **1**(4), August 1993.

[13] S. Floyd and K. Fall, Router Mechanisms to Support End-to-End Congestion Control, Technical report, Network Research Group at LBNL, 1997.

[14] S. Floyd and K. Fall, Promoting the use of end-to-end congestion control in the Internet, *IEEE/ACM Transaction on Networking*, August 1999.

[15] C. Fraleigh, S. Moon, C. Diot, B. Lyles, F. Tobagi, Architecture of a Passive Monitoring System for IP Networks, *Sprint technical report TR00-ATL-1018*, `http://www.sprintlabs.com`, October 2000.

[16] Ganymede Software, Chariot 3.2, March 2000, `http://www.ganymedesoftware.com`.

[17] C. Hollot, V. Misra, D. Towsley, W. Gong, A control theoretic analysis of RED, in *Proceedings of IEEE Infocom*, 2001.

[18] V. Jacobson, Congestion avoidance and control, in *Proceedings of ACM Sigcomm*, 1988.

[19] D. Lin and R. Morris, Dynamics of Random Early Detection, in *Proceedings of ACM Sigcomm*, 1997.

[20] R. Morris, TCP Behavior with Many Flows, in *Proceedings of IEEE/ICNP*, 1997.

[21] T. J. Ott, T. Lakshman, and L. Wong, SRED: Stabilized RED, in *Proceedings of IEEE Infocom*, 1999.

[22] K.K. Ramakrishnan and S. Floyd, A Proposal to add Explicit Congestion Notification (ECN) to IP, RFC 2481, January 1999

[23] L. Rizzo, Dummynet: a simple approach to the evaluation of network protocols, *ACM Computer Communication Review*, January 1997.

[24] V. Rosolen, O. Bonaventure, and G. Leduc, A RED discard strategy for ATM networks and its performance evaluation with TCP/IP traffic, *ACM Computer Communication Review* 29(3), July 1999.

[25] T. Ziegler, C. Brandauer, and S. Fdida, A quantitative model for parameter setting of RED with TCP traffic, to appear in *Proceedings of IWQoS*, 2001.

APPENDIX

A. RED PARAMETERS

In [17], the authors propose the following condition to be satisfied in order to guarantee a stable response of RED:

$$\frac{L_{red}(R^{+}C)^{3}}{(2N)^{2}} \leq \sqrt{\frac{w_g^2}{K^2}+1} \tag{1}$$

where R^{+} is the maximum round trip time, C is the capacity of the link (in packets/sec), N is the number of flows, while L_{red}, K and w_g are defined as follows:

$$L_{red} = \frac{max_p}{max_{th} - min_{th}} \tag{2}$$

$$K = \frac{\log_e(1-\alpha)}{\delta} \tag{3}$$

$$w_g = 0.1 \min\left\{\frac{2N}{(R^{+})^2 C}, \frac{1}{R^{+}}\right\} \tag{4}$$

In our testbed the network parameters are: $N = 256$, $C = 830$ packets/sec[7], $max_{th} = 130$, $min_{th} = 30$, $max_p = 0.1$, $R^{+} = 296$ ms[8], $\alpha = 0.002$, $\delta = \frac{1}{C} = 0.0012$.

Then, from (2), $L_{red} = 10^{-3}$, from (3), $K = 1.668$ and from (4) $w_g = 0.338$.

Substituting these values in (1), we can see that the RED parameters as defined in Section 3.2 satisfy the requirement proposed in [17].

[7]The link capacity is 10Mbps and the packet size is 1500 bytes.
[8]The target queue occupancy is set to $\frac{max_{th}+min_{th}}{2}$