# HRED: A Simple and Efficient Active Queue Management Algorithm

Liujia Hu
Cisco Systems
Email: ljhu@cisco.com

Ajay D. Kshemkalyani
Univ. of Illinois at Chicago
Email: ajayk@cs.uic.edu

***Abstract*—Active Queue Management (AQM) is an area of critical importance for the operation of networks. In this paper, we propose a minimal adjustment to the classic Random Early Detection (RED) algorithm, called Hyperbola RED (HRED), that uses the Hyperbola as the drop probability curve. The control law of HRED can regulate the queue size close to the reference queue size which is settable by the user. As a result, it is expected that HRED is no longer sensitive to the level of network load, its behavior shows low dependence on the parameter settings, and it can achieve higher network utilization. Additionally, very little work needs to be done to migrate from RED to HRED on Internet routers because only the drop profile is adjusted. We implemented HRED on a real Internet router to examine and compare its performance with the classic RED and Parabola RED that are currently deployed on Internet routers. From experiments on the real network, we conclude that HRED is insensitive to the network conditions and parameter settings, and can achieve higher network utilization than the other RED schemes.**

## I. Introduction

As a congestion control mechanism, the Random Early Detection (RED [11], hereafter called classic RED) algorithm has been proposed and widely implemented on Internet routers for many years. It can provide the Internet with the ability to absorb bursts and with lower-delay interactive service. Many studies have tried to enhance this algorithm for today's Internet community. This has led to a very active research area called Active Queue Management (AQM) which focuses on the congestion control problem. The AQM mechanism is recommended on Internet routers to achieve the following goals: managing queue lengths to absorb short-term congestion (e.g., bursts), providing lower end-to-end interactive delay, and avoiding global synchronization phenomena [3].

The RED algorithm functions by detecting incipient congestion and notifying TCP by marking or dropping (hereafter using the term "drop"[1]) packets probabilistically before the queue in a router fills up. Briefly, the algorithm works by maintaining a running average queue size. As the average queue size varies between the minimum ($min_{th}$) and maximum ($max_{th}$) threshold, the probability of dropping a packet changes linearly between zero and the maximum drop probability value ($max_p$). Thus, the drop probability curve is linear to the change of the average queue size. If the average queue size exceeds $max_{th}$, all arriving packets are dropped [11]. Since the drop mechanism is based on the moving average algorithm, RED is able to control the transient congestion by absorbing arrival rate fluctuations.

In the past few years, many enhanced algorithms or RED variants have been proposed to overcome some weaknesses of the classic RED algorithm, leading to very active AQM research. Differently than in [12], we classify these algorithms into two categories, based on their approach.

- The algorithms in the first category apply control theory to model TCP with RED dynamics, and then develop new feedback mechanisms and control laws to achieve an optimal and stable operating point. Examples of such algorithms are [8], [12], [16], Proportional Integrator [15], [14], [13], and PIP [25]. This category focuses on the design of a more stable and faster responsive control system. For example, in [14], [15], the low-pass filter design of the classic RED algorithm is believed to lead to instability and low frequency oscillations in the regulated output from a control standpoint. Therefore, the authors proposed a stabilizing proportional controller, called PI, that uses instantaneous samples of the queue size to overcome these drawbacks.
- The algorithms in the second category use approaches that do not use control theory. Examples of such algorithms are Gentle RED [10], Adaptive RED [7], Stabilized RED [21], Flow RED [18], Balanced RED [1], non-linear RED [22], and Random Exponential Marking (REM) [2]. These algorithms aim to improve some or all of the following features: fairness, network utilization, packet loss, and adaptability for different characteristics of traffic/load. For instance, Adapt RED, a self-tuning algorithm, dynamically adjusts its $max_p$ in response to the condition of network congestion reflected by the average queue size [7].

Although the above body of research has contributed many insights into congestion control mechanisms, the algorithms therein also have their own technical drawbacks. Furthermore, they suffer from the following practical limitations. The first limitation is the lack of feasibility of the implementation on Internet routers. Of the many AQM algorithms, only the following have been implemented by the leading Internet router

[1]IETF proposed a standard called Explicit Congestion Notification (ECN) [23]. With ECN, an AQM scheme can signal congestion to the end-system by marking instead of dropping a packet. This topic is not in the scope of this paper.

vendors: "Parabola RED" by Cisco Systems [6], [16], and "Gentle RED"-like by Juniper networks [17]. AQM algorithms are often complex to implement because it is not simply a matter of "computational complexity" but rather of "packet forwarding engine design" in its totality. Any new AQM implementation significantly impacts the router architecture design as well as the software flows. The second limitation is the lack of operational guidance for these AQM schemes on real networks.

The primary goal of our work is to design a simple and practical RED variant that is able to maintain the queue size to the target value under a wide range of network loads. As a result, the queue size should no longer be sensitive to the level of congestion. As has been pointed out in [7], [19], [20], one major drawback of the classic RED is that the queue size varies with the level of congestion. This also implies an unpredictable queueing delay from RED, that has negative impacts on the network design. Thus, to overcome this issue, this paper proposes a minimal adjustment to the classic RED algorithm, called Hyperbola RED (HRED), which uses the Hyperbola as the drop probability curve (Fig. 1, equations are after transforming the coordinates). The control law of HRED can regulate the queue size to the reference queue size (i.e., the maximum threshold $max_{th}$). This reference queue size is settable by the user by specifying the maximum threshold $max_{th}$. As a result, HRED is no longer sensitive to the level of network load, and the queueing delay from HRED becomes predictable. Furthermore, very little work needs to be done to migrate from RED to HRED on Internet routers because only the drop profile is adjusted.

Another goal of our work is to study the link utilization from the perspective of the network layer (Section IV). We define the network utilization as the ratio of the departure traffic rate to the arrival traffic rate on the router; this ratio is measurable. The observation in this paper is that from the viewpoint of the network layer, the queue size is a decisive influence on the network utilization. More specifically, when an AQM scheme such as RED is deployed on Internet routers, while a network can provide lower-delay service with a lower queue size, its network utilization also lowers. Although the traditional queue management scheme of Tail Drop (TD) removes the ability to provide lower-delay service, it can provide the network with higher utilization. For a given set of RED parameters, HRED can achieve higher network utilization than other RED schemes because it maintains the queue size close to $max_{th}$. Furthermore, by adjusting the reference queue size (i.e., $max_{th}$), HRED can achieve the optimal point for the tradeoff between delay and the network utilization.

From our experiments and analysis, we conclude that HRED has many significant advantages compared to all other schemes that are currently deployed on Internet routers.

- The control law of HRED can regulate the queue size to the reference queue size (i.e., the maximum threshold $max_{th}$) under various network conditions. This reference queue size is settable by the user (by specifying the maximum threshold $max_{th}$). As a result, HRED is no longer sensitive to the level of network load.
- HRED is less sensitive to the parameter settings than other schemes. For a given $max_{th}$, HRED behaves similarly (does not depend much on other parameter settings).
- HRED can achieve higher network utilization.
- HRED can result in predictable average queueing delays of networks.
- HRED retains the ability to control the short-term congestion by absorbing bursts, because it still keeps the moving average queue size algorithm and maintains a non-full queue. The queue size is settable and depends on the requirement.
- HRED is simple and can be easily implemented on Internet routers – only the drop profile is adjusted compared to the classic RED algorithm.
- An additional benefit is that by adjusting the reference queue size (i.e., $max_{th}$), HRED helps network administrators tune to an optimal point for the tradeoff between delay and the network utilization.

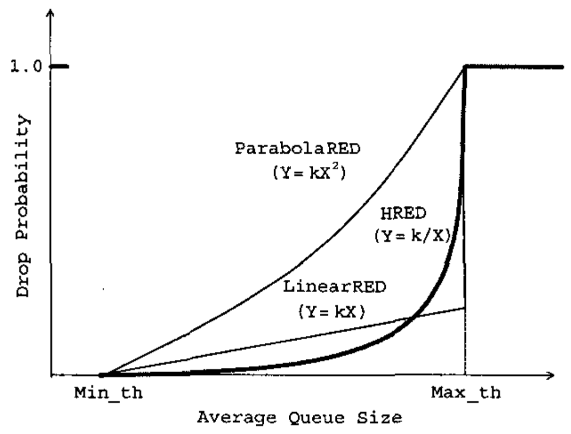

Fig. 1. Hyperbola as drop probability curve

The rest of this paper is organized as follows. In Section II, we describe our experimental methods including our experimental network, traffic generation, and the parameter settings for RED. In Section III, we discuss how RED works and experimentally demonstrate its advantages in queuing behavior, including (i) its insensitivity to the level of congestion, and (ii) the settable target queue size. In Section IV, we study the network utilization of HRED. Section V gives the conclusions and directions for future work.

## II. Experimental Methods

### A. Experimental Network

To ensure our experiments and conclusions are accurate and based on realistic network environments, instead of using simulations, we set up a testbed with Cisco 12000 series high-end Internet routers. We implemented Hyperbola RED on Cisco routers based on IOS 12.0. The experimental network is illustrated in Fig. 2. This is a typical Provider Edge (PE) router to Customer Edge (CE) router topology.

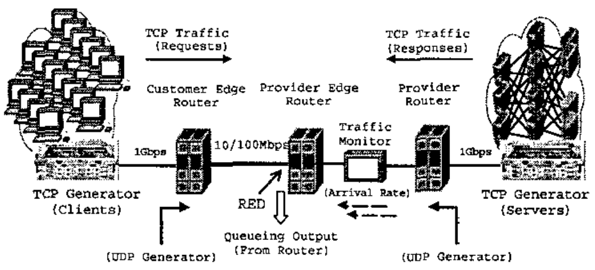

Fig. 2. Experimental network topology

Congestion occurs when there is contention for bandwidth. This happens in the following three types of network nodes: the node of aggregation, the node of speed mismatch, and the "LAN to WAN" interface node. Our experimental environment actually replicates the node of aggregation and the node of speed mismatch: the ISP sells bandwidth to customers and also provides Internet services. Customers can only get the bandwidth that they pay for. Because of the much higher bandwidth of the ISPs' backbone links, the PE router must restrict its egress interfaces to customers and provision the corresponding bandwidth. Normally, customers do not fully use their bandwidth all the time, but during the peak time, oversubscription often happens. Once the requested bandwidth by customers is greater than the offered bandwidth of the PE router, congestion occurs on this bottleneck link between PE and CE router. Therefore, an AQM algorithm such as RED is usually enabled for the purpose of congestion control on the interface of the PE router facing the CE router. Cisco high-end routers (12000 series) are used in the experiments. Two 4xOC12 POS Internet Service Edge (ISE) linecards on the PE and CE router are connected as the bottleneck link. In our experiments, we restrict (by shaping) this link as a 10Mbps or 100Mbps bottleneck (because 10Mbps and 100Mbps are popular bandwidth types). The queueing behavior is recorded by the router on which RED is enabled (i.e., PE router).

Another critical setup in our experimental environment is the "Traffic Monitor" (Adtech) seated on the backbone. Because all traffic flows have to go through this traffic monitor from the core to the edge, it can provide real-time arrival rate of aggregate traffic, or of a particular traffic flow. We use this setup to examine how AQM schemes control the long-term persistent congestion (Section IV).

### B. Traffic Generation

Short-lived and long-lived TCP flows have very different characteristics. Short-lived TCP flows seldom operate beyond the slow-start phase of their transmission, and therefore their window size is generally smaller and they always generate smaller packet bursts. Long-lived TCP connections mostly operate in congestion avoidance phase with larger windows, and they produce larger packet bursts. While Internet traffic is dominated by short-lived flows, the majority of the bytes in the Internet belong to long-lived flows [24], [26]. Long-lived (FTP-like) TCP traffic is used to generate the persistent congestion in each of our experiments. Considering that the Internet is also a heterogeneous environment, the mix of TCP and UDP traffic is also used in some of our experiments. TCP flows are generated by the equipment called Avalanche. Avalanche is a traffic generator that can replicate real-world TCP applications including HTTP (1.0/1.1), FTP, SMTP, Telnet, DNS, and so on. One Avalanche on the customer side is used to create a large array of PCs with streaming clients, and the other one on the provider's side is used to create an array of servers. Thus, in this client-server mode in our experiments, the clients send out request traffic while the servers send back response traffic. This TCP traffic generator also provides granular control: TCP paramters such as the Maximum segment size, Receive window, Retries, and so on, are all settable. In our experiments, the Maximum segment size is set to 1460 bytes, the Receive window is set to 32768 bytes, and Retries set to 10. Slow start and congestion control are enabled. By adjusting the object size, we can obtain short-lived or long-lived TCP traffic. UDP traffic is generated by the UDP traffic generator (Agilent). The UDP packet size is set to 512 bytes.

### C. RED and Parameter Settings

The RED algorithm uses the following parameters.

- $w_q$: weighting factor for average queue size calculation.
- $max_p$: maximum drop probability value.
- $min_{th}$: minimum threshold of the average queue size beyond which packets are probabilistically dropped.
- $max_{th}$: maximum threshold of the average queue size beyond which packets are forcibly dropped.

Although it has turned out to be difficult to find a unique setting of the parameters that works for all conditions, many studies and realistic deployment have contributed guidance on how to set configuration parameters [9], [27].

Recently, to overcome some well known weaknesses of the classic RED, Jacobson *et al.* worked with the network gear industry to propose a RED variant, called Parabola RED (parabola as the drop profile), and implemented it on real Internet routers [6], [16]. In [16], results show Parabola is a better AQM scheme than the classic RED (or linear RED). Therefore, in all our experiments, we compare HRED with the Parabola RED (since no other AQM schemes are implemented on today's Internet routers).

On the basis of an extensive investigation, the industry tends to set RED parameters as follows [5]:

$$w_q = 1/512,\ max_p = 1.0,\ min_{th} = 0.03 * B$$
$$max_{th} = 3 * min_{th} \approx 0.1 * B$$

where $B$ is the output link bandwidth in MTU-sized (Maximum Transmission Unit) packets. The formula is:

$$B = Linerate/(8bits/byte)/MTU$$

For example, for a 10Mbps link with MTU equal to 1500 bytes, $B$ can be calculated as:

*B = 10Mbps / (8bits/byte) / 1500 (bytes/packet) = 833*

Then $min_{th}$ is roughly 25 packets, and $max_{th}$ is roughly 84 packets[2]. Also, for a 100Mbps link with MTU equal to 1500

[2]Due to some hardware restrictions, the difference between $max_{th}$ and $min_{th}$ has to be power of 2. Hence, we slightly adjust $max_{th}$ to 89 packets.

bytes, the $min_{th}$ is roughly 250 packets, and $max_{th}$ is roughly 834 packets.

For all our experiments, we use a dedicated physical buffer of size within 1.25 to 2 times the bandwidth-delay product [4]. This size is 200 packets for a 10Mbps network and 2000 packets for a 100Mbps network. Wherever not explicitly mentioned, a 10Mbps network (bottleneck link) is also assumed.

## III. HYPERBOLA RED (HRED)

One major drawback of the classic RED is that it is quite sensitive to the level of congestion and to the RED parameter settings. In other words, the resulting queue size varies with the level of congestion and with the parameter settings. This also implies an unpredictable queueing delay from RED, that has negative impacts on the network design. Thus, to overcome these issues, many RED variants have been proposed. Among them, Parabola RED, proposed by Jacobson *et al.* and implemented on Internet routers, uses a quadratic curve as the drop profile. This is believed to work for a much wider variety of traffic and link bandwidths [6], [16]. However, while it results in an improvement over the classic RED, it still has several drawbacks.

More recently, researchers applied control theory to the design of AQM mechanisms. Control is applied to the queue size, stability, queue utilization, and so on [8], [12], [14], [15]. Simulations suggest some improvements. However, as mentioned in Section I, the lack of feasibility of the implementation becomes a barrier for real-world deployment.

In this section, we demonstrate by using experimentation that Hyperbola RED, which requires a minimal change to the classic RED algorithm, is able to overcome various drawbacks of the existing algorithms. Because HRED uses the Hyperbola as the drop probability curve, this kind of control law can regulate the queue size to the reference queue size (i.e., the maximum threshold $max_{th}$) under a much wider range of traffic loads. In other words, HRED is insensitive to the level of network load. As a result, the queueing delay is less unpredictable, because the queue size does not vary much with the level of congestion. We also see that HRED is insensitive to the parameter settings.

### A. Experiment 1

Fig. 3 shows the insensitivity of HRED to the level of congestion. We increase the congestion by stepping the number of FTP flows from 0 to 4500. Then we decrease the congestion. Both in HRED and Parabola RED, $min_{th}$ = 25 packets and $max_{th}$ = 89 packets, $w_q$ = 1/512 and $max_p$ = 1.0, as calculated in Section II-C (RED parameter settings). As seen from the figure, HRED rapidly reaches and keeps its reference queue size (i.e., $max_{th}$), irrespective of whether the congestion is increased or decreased. However, Parabola RED is quite sensitive to the level of congestion.

Fig. 4 also demonstrates that HRED is insensitive to the traffic load, from another perspective. These are the comparisons between HRED and Parabola RED in terms of the arithmatic average queue size as a function of the number of TCP flows. HRED is much less sensitive to the level of network loads.

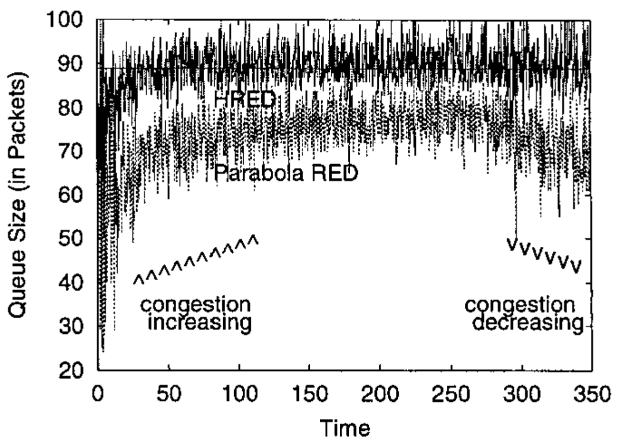

Fig. 3. Expt. 1a) Insensitivity to level of congestion: HRED vs. Parabola RED

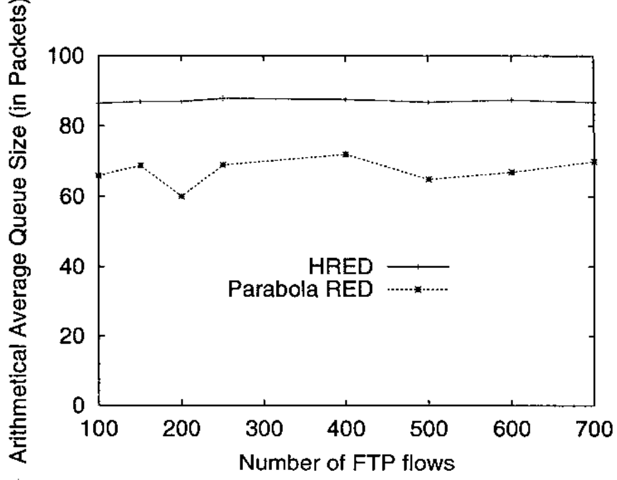

Fig. 4. Expt. 1b) Arithmatic average queue size under 100 - 700 FTP flows, $min_{th}$ = 25 & $max_{th}$ = 89

### B. Experiment 2

Fig. 5 displays the insensitivity of HRED to thè parameter settings. Given a reference queue size (maximum threshold $max_{th}$), HRED results in a very similar queueing behavior as $min_{th}$ is varied. Fig. 6 shows that Parabola RED behaves quite differently with different parameter settings of $min_{th}$ (even though $max_{th}$ is the same).

We note that Jacobson et al. [16] concluded that of the 4 parameters of RED, $max_p$ and $w_q$ do not play a critical role, so their values tend to be fixed in industry. Therefore in our experiment, we tested only the sensitivity to $min_{th}$ and $max_{th}$.

### C. Experiment 3

Another advantage of HRED is that the reference queue size is settable by the user, by adjusting $max_{th}$. As can be seen from Fig. 7, this results in predictable average delays of networks. Network administrators can use this information to tune networks to an optimal operating point. An additional benefit will be discussed in Section IV – by adjusting the

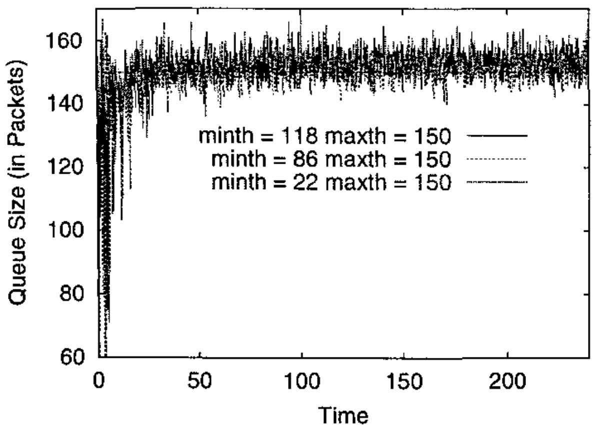

Fig. 5. Expt. 2a) Insensitivity of HRED to parameter settings for the given $max_{th}$

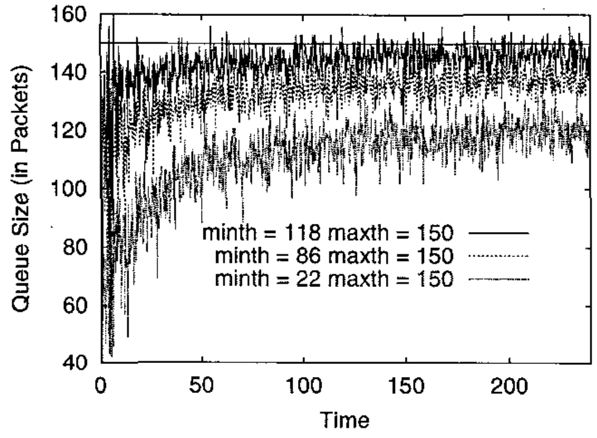

Fig. 6. Expt. 2b) Sensitivity of Parabola RED to parameter settings for the given $max_{th}$

reference queue size, network administrators can decide the tradeoff between the delay and network utilization.

## IV. Network Utilization

Routers at a bottleneck tend to be congested when the packets are incoming at a rate that exceeds the capacity of a given link. In other words, congestion occurs whenever the interface arrival rate exceeds the interface departure rate. Congestion may persist over different time-scales. We define the "long-term persistent congestion" as the congestion events that last much longer than a round-trip time. Under long-term persistent congestion, the primary goal of the congestion control mechanism at the bottleneck is to regulate the arrival rate of the aggregate traffic to best conform to the capacity of the given link. Otherwise, packet drops at the bottleneck could result in great waste of network resources, because all the resources a packet has consumed in transit get wasted when it is dropped before reaching its destination. This is further aggravated as TCP senders are normally spread across the whole network. Therefore, it is very crucial for AQM schemes to regulate TCP flows from the sending point (i.e., the source of a TCP flow). Because the router works in the network layer,

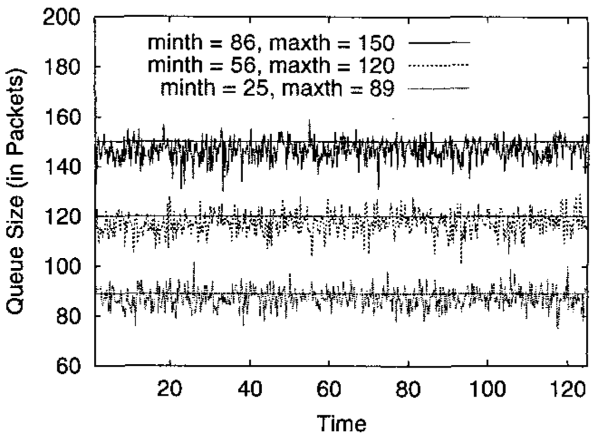

Fig. 7. Expt. 3) HRED: Reference queue size is settable by the user. 600 FTP flows

we can define the "Network Utilization" as the departure rate (which equals the link capacity during congestion) over the arrival rate, i.e., $U$ = departure rate / arrival rate. The closer a queue management scheme regulates the arrival rate to the link capacity, the higher the network utilization it achieves.

AQM schemes always maintain a lower queue size for the lower-delay service and for transient congestion control. However, these benefits are not without cost. A lower queue size leads to a lower network utilization. Thus, while a network can provide lower-delay service, its network utilization also lowers. Although the traditional queue management scheme of Tail Drop (TD) removes the ability to provide lower-delay service, it can provide higher network utilization due to the larger queue size (full queue). For a given set of RED parameters, HRED can achieve higher network utilization than other RED schemes, because it maintains the queue size to $max_{th}$. Furthermore, by adjusting the reference queue size (i.e., $max_{th}$), HRED can achieve the optimal point for the tradeoff between delay and the network utilization.

### A. Experiment 4

Fig. 8 gives the comparison of all schemes (classic RED, Parabola RED, Hyperbola RED, and Tail Drop), under a load of 50 FTP flows, 10Mbps bottleneck link, $max_{th} = 89$. It is apparent that TD best regulates the aggregate traffic arrival rate to conform to the capacity of the bottleneck line (but with biases against flows and global synchronization). TD regulates the arrival rate of the aggregate traffic to about 10.7Mbps, so the network utilization is around 93.5%. Therefore, TD offers the highest network utilization from this point of view. Among the three AQM schemes, Hyperbola RED provides the highest network utilization due to a larger queue size, Parabola RED performs the second, and the linear RED (classic RED) performs the worst due to the lowest queue size (about 45 packets). On the other hand, linear RED can provide the lowest delay, while TD results in the largest queueing delay.

Fig. 9 gives an analogous comparison of all schemes under even heavier congestion (created by a load of 150 FTP flows, 10Mbps bottleneck link) and $max_{th} = 89$. However, in this

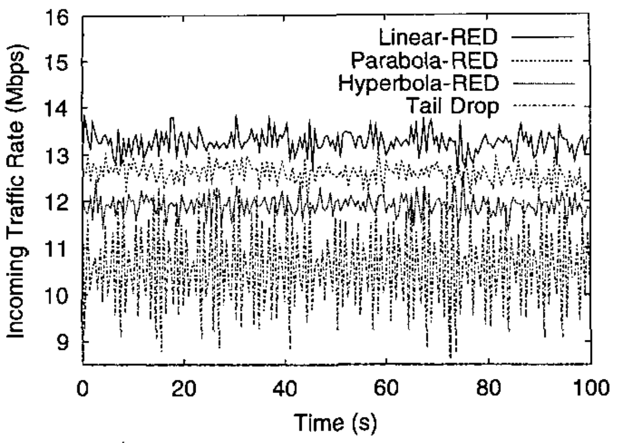

Fig. 8. Expt. 4a) Comparing all schemes, 50 FTP flows, 10Mbps bottleneck

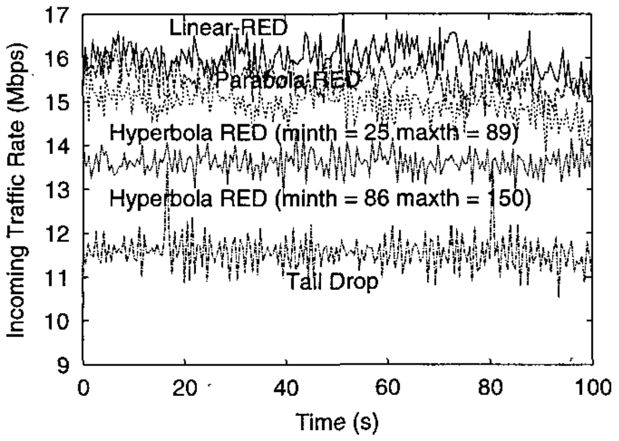

Fig. 9. Expt. 4b) Comparing all schemes, 150 FTP flows, 10Mbps bottleneck

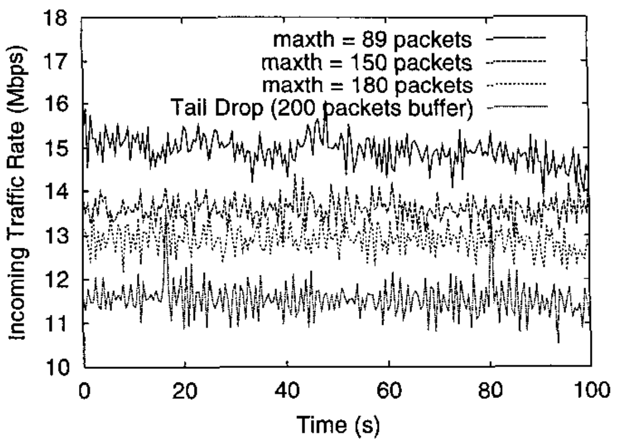

Fig. 10. Expt. 4c) HRED with different $max_{th}$ values, 150 FTP flows, 10Mbps bottleneck

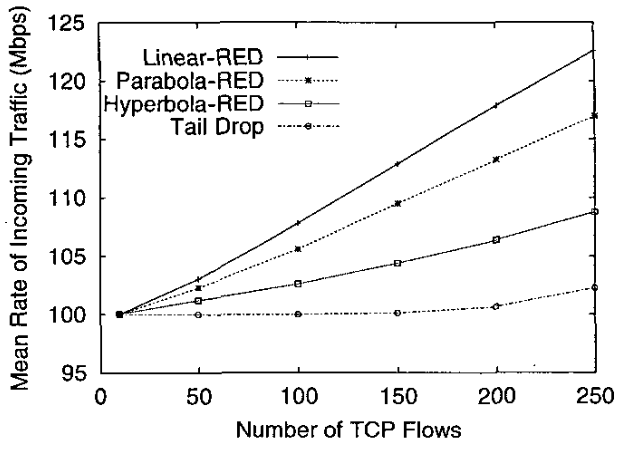

Fig. 11. Expt. 5) All schemes, mean values of aggregate traffic arrival rate, different FTP loads, 100Mbps bottleneck

figure, we specifically add one more test for Hyperbola RED with a higher $max_{th}$ value equal to 150 packets. We can see that while a bigger queue size leads to a bigger delay, it is better than others in terms of network utilization.

Fig. 10 shows the result of the experiment using HRED to study the effect of $max_{th}$ on the utilization. We tested 3 scenarios, i.e., HRED with 3 different $max_{th}$ values, and under 150 FTP flows and using a 10Mbps bottleneck link. It is clearly seen that a higher $max_{th}$ results in a better utilization.

### *B. Experiment 5*

To further investigate the performance of AQM schemes under a wider range of loads, Fig. 11 plots the mean values (arithmetic average) of the aggregate traffic arrival rate under 10, 50, 100, 150, 200, and 250 FTP flows for the 100Mbps bottleneck link. We observed that TD has the advantage in getting higher network utilization. However, for TD, we also observed the biases against flows as well as the global synchronization (plus the largest delay).

### *C. Experiment 6*

Due to the proliferation of streaming media content over the Internet, unresponsive traffic and real-time traffic (such as UDP-based streams) form a significant portion of today's Internet traffic. Real-time traffic is not necessarily responsive to network congestion, so it is important to examine the behavior of a heterogeneous environment. In this experiment, 10 percent of the bottleneck bandwidth was UDP-based traffic.

- Fig. 12 plots how the different schemes control the long-term network congestion under 150 FTP flows and 1Mbps UDP-based traffic.
- Fig. 13 plots another perspective (the mean values of the arrival rate as a function of the number of TCP flows) for all schemes under a TCP and UDP traffic mix.

Compared to a pure TCP environment, all congestion control methods are less efficient in an environment with a mix of TCP and UDP. This is because UDP-based traffic is unresponsive to packet drops, whereas TCP-based traffic is. This is expected as per [3]. The UDP-based traffic is competing with TCP for bandwidth, and this equals the load increase to the bottleneck, which makes the congestion even heavier.

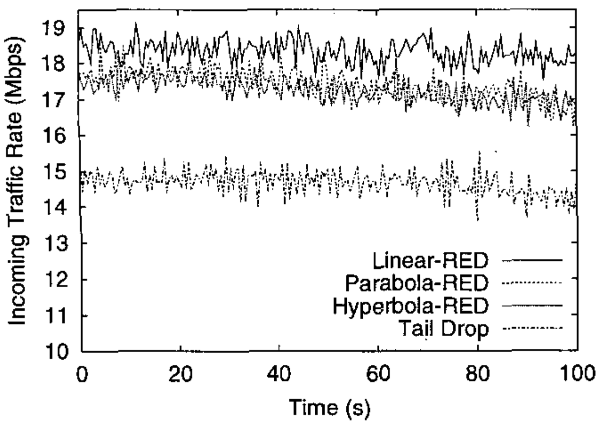

Fig. 12. Expt. 6a) All schemes, 150 FTP flows plus 1Mbps UDP, 10Mbps bottleneck

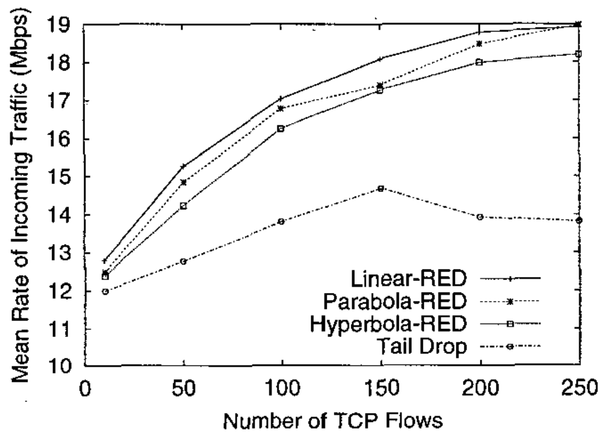

Fig. 13. Expt. 6b) All schemes, mean value of arrival rate under different FTP loads plus 1Mbps UDP, 10Mbps bottleneck

## V. Conclusions and Future Work

Although many AQM schemes have been proposed, few of them have actually been deployed on today's Internet router by the industry due to various reasons as mentioned earlier. At present, only the classic RED algorithm and its two variants, Gentle-RED and Parabola RED, are deployed. In this paper, we proposed a minimal adjustment to the classic RED algorithm, called Hyperbola RED (HRED), that uses the Hyperbola as the drop probability curve. It is a simple but efficient AQM algorithm. Although HRED still suffers from the slow responsive issue (due to the moving average queue calculation), from our experiments and analysis, we concluded that HRED has many advantages over existing active queue management algorithms (the advantages were listed in Section I). In this paper, we also defined and studied the network utilization of AQM schemes, from the perspective of the network layer. HRED was seen to have various advantages, which were presented in Section IV.

For further work, we are interested in studying HRED with ECN. A lot of research has proved that AQM with ECN performs more efficiently than without ECN. We are also interested in studying HRED using short-lived TCP (Web-like) traffic.

## References

[1] F. Anjum and L. Tassiulas, Fair Bandwidth Sharing among Adaptive and Non-Adaptive Flows in the Internet, *Proc. IEEE INFOCOM'99*, 1999, p. 1412-1420.

[2] S. Athuraliya, S. Low, V. Li, and Q. Yin, REM: Active Queue Management, *IEEE Network*, vol. 15(3), 2001, p. 48–53.

[3] B. Braden et al. Recommendations on Queue Management and Congestion Avoidance in the Internet, *RFC 2309*, 1998.

[4] M. Christiansen, K. Jeffay, D. Ott, and F.D. Smith, Tuning RED for Web Traffic, *Proc. ACM SIGCOMM 2000*, 2000, p. 139-150.

[5] Cisco Systems, IOS Configuration Guide, WRED on the Cisco 12000 Series Router. *http://www.cisco.com/univercd/cc/td/doc/product/software/ios112/ios112p/gsr/wred_gs.htm*

[6] Cisco Systems. White Paper: Using Persistence vs. Instantaneous Queue Depth in RED Implementation, *http://www.cisco.com/warp/public/cc/pd/rt/12000/tech/queue_wp.htm*

[7] W. Feng, D. Kandlur, D. Saha, K. Shin, A Self-Configuring RED Gateway, *Proc. IEEE INFOCOM'99*, 1999, p. 1320-1328.

[8] V. Firoiu and M. Borden, A Study of Active Queue Management for Congestion Control, *Proc. IEEE INFOCOM (3)*, 2000, p. 1435-1444.

[9] S. Floyd, References on RED Queue Management, *http://www.icir.org/floyd/red.html*, April 2002.

[10] S. Floyd and K. Fall, Promoting the Use of End-to-End Congestion Control in the Internet, *IEEE/ACM Transactions on Networking*, 7(4), Aug. 1999, p. 458-472.

[11] S. Floyd and V. Jacobson, Random Early Detection Gateways for Congestion Avoidance, *IEEE/ACM Transactions on Networking* 1(4), Aug. 1993, p. 397-413.

[12] Y. Gao, J. Hou, A State Feedback Control Approach to Stabilizing Queues for ECN-Enabled TCP Connections, *Proc. IEEE INFOCOM 2003*, 2003, p. 2301-2311.

[13] C.V. Hollot, Y. Liu, V. Misra, and D. Towsley, Unresponsive Flows and AQM Performance, *Proc. IEEE INFOCOM 2003*, 2003, p. 85-95.

[14] C. Hollot, V. Misra, D. Towsley, and W. Gong, A Control Theoretic Analysis of RED, *Proc. IEEE INFOCOM 2001*, 2001, p. 1510-1519.

[15] C. Hollot, V. Misra, D. Towsley, and W. Gong. On Designing Improved Controllers for AQM Routers Supporting TCP Flows, *Proc. IEEE INFOCOM 2001*, 2001, p. 1726-1734.

[16] V. Jacobson, K. Nichols, and K. Poduri, RED in A Different Light. Technical Report, Cisco Systems, September 1999.

[17] Juniper Networks, White Paper: Supporting Differentiated Service Classes: Active Queue Memory Management, Feb 2002. *http://www.juniper.net/solutions/literature/white_papers/200021.pdf*

[18] D. Lin and R. Morris, Dynamics of Random Early Detection, *Proc. ACM SIGCOMM'97*, 1997, p. 127-137.

[19] M. May, J. Bolot, C. Diot, and B. Lyles. Reasons not to deploy RED, *Proceedings of 7th International Workshop on Quality of Service (IWQoS '99)*, June 1999, p. 260-262.

[20] V. Misra, W.-B. Gong, D. Towsley, Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED, *Proc. ACM SIGCOMM 2000*, 2000, p. 151-160.

[21] T. J. Ott, T. Lakshman, and L. Wong, SRED: Stabilized RED, *Proc. IEEE INFOCOM 1999*, 1999, p. 1346-1355.

[22] E. Plasser, T. Ziegler, and P. Reichl, On the Non-Linearity of the RED Drop Function, *Proceedings of International Conference on Computer Communication (ICCC)*, Mumbai, August 2002.

[23] K. Ramakrishnan, S. Floyd, D. Black, The Addition of Explicit Congestion Notification (ECN) to IP, *RFC 3168*, September 2001.

[24] K. Thompson, G. Miller, and R. Wilder, Wide-Area Internet Traffic Patterns and Characteristics, *IEEE Network*, 11(6), Nov. 1997, p. 10-23.

[25] H. Zhang, B. Liu, W. Dou, Design of a Robust Active Queue Management Algorithm Based on Feedback Compensation, *Proc. ACM SIGCOMM'03*, 2003.

[26] Y. Zhang and L. Qiu, Understanding the End-to-End Performance Impact of RED in a Heterogeneous Environment, Cornell CS Technical Report 2000-1802, July 2000.

[27] T. Ziegler, C. Brandauer, and S. Fdida, A Quantitative Model for Parameter Setting of RED with TCP Traffic, *9th International Workshop on Quality of Service (IWQoS)*, Karlsruhe, June 2001, p. 202-216.