복합식 Random Early Detection 알고리즘

주 창 희*, 정회원 박 세 웅**

Hybrid Random Early Detection

Chagnhee Joo*, Saewoong Bahk** *Regular Members*

요 약

본 논문에서는 새로운 AQM (Active Queue Management) 알고리즘으로, RED (Random Early Detection) 알고리즘을 기반의 복합식 RED (HRED) 알고리즘을 제안한다. HRED는 큐 길이를 관리하는 알고리즘과 동적인 트래픽에 적응하는 알고리즘을 분리하여, 낮은 손실률과 전송 지연을 가지면서도 네트워크의 활용도를 높일 수 있도록 고안되었다. 분석적인 방법을 통해, HRED 알고리즘의 안정성을 증명하고 빠른 응답속도를 가지는 설정을 제안한다. 모의실험을 통해, HRED가 안정성 및 응답속도의 측면에서 다른 이전의 AQM 알고리즘보다 개선된 성능을 가지며, 간단한 설정을 통해 정적이거나 동적인 트래픽 상황에 잘 적응함을 보인다.

ABSTRACT

In this paper, we propose a new AQM algorithm, Hybrid RED (HRED). It combines the more effective elements of recent algorithms with a RED core. It decouples queue length management from adaptation to dynamic load in order to achieve high network utilization with low loss and delay in a simple and scalable manner. We analyze it to prove local stability and provide a simple configuration for quick response time. Throughout simulation, we demonstrate that HRED outperforms earlier AQM algorithms in stability and response time with straightforward selection of parameters for both steady load and changes in load.

I. Introduction

AQM algorithms help end hosts make decisions about transmission rates by providing congestion information based on characteristics of a router's queue. The advantages of such feedback seem obvious, and the IETF recommends the use of AQM to reduce loss rate, to support low-delay interactive services, and to avoid TCP lock-out behavior[1]. A successful AQM algorithm must provide simple, intuitive means of configuring parameters to achieve stable operation under steady load and rapid adaptation to changes in load. In addition, an algorithm should be intuitive and adequate for achieving desired goals. Without those capability, the algorithm may remain unused.

Numerous algorithms, such as RED[2], Adaptive RED[3], Blue[4], the Adaptive Virtual Queue (AVQ)[5], and the PI controller[6], are proposed for AQM, but they have still difficulties in configuring for stable operation in dynamic networking environments, which prevented them from replacing the traditional tail-drop mechanism in the Internet.

In this paper, we introduce and evaluate a new AQM algorithm that combines a RED design with aspects of other AQM algorithms, such as Blue and the Proportional controller[6]. For this reason, we call it Hybrid RED, or HRED. HRED

* 서울대학교 전기공학부 뉴미디어통신공동연구소 데이터통신망연구실 (cjoo@netlab.snu.ac.kr),
** 서울대학교 전기공학부 뉴미디어통신공동연구소 데이터통신망연구실 (sbahk@netlab.snu.ac.kr)
※ 본 논문은 2002년 [illegible] 우수논문으로 선정되어 게재 추천된 논문입니다.

provides intuitive configurations and rapid response as well as stable operation.

The remainder of the paper is organized as follows. We begin with an overview of previous work in Section II. In Section III, we provide design principles and a simple analysis of HRED. After evaluating HRED and other AQM algorithms through simulation in Section IV, we conclude in Section V.

Ⅱ. Previous Work

RED[1] uses two operations to calculate drop probability: it first computes an average queue length $\bar{l}$ using an Exponential Weighted Moving Average (EWMA) with weight w_q, then calculates drop probability p from $\bar{l}$ using a linear mapping function, which has three parameters: min_{th}, max_{th}, and p_{max}. Average queue lengths in the operating range $[min_{th}, max_{th}]$ are linearly mapped to drop probabilities in the range $[0, p_{max}]$. Below min_{th}, packets are not dropped, and above max_{th}, they are always dropped. The simplest modification of RED to improve stability is to use the **gentle_ option**[7], which eliminates the discontinuity at max_{th} and makes the drop probability function piecewise linear by mapping the interval between the maximum threshold and the buffer size, $[max_{th}, Q_{max}]$, linearly to the range $[p_{max}, 1]$.

Rather than expanding the linear range, Adaptive RED (ARED) adjusts p_{max}[3]. It increases p_{max} when $\bar{l}$ gets over max_{th} and decreases when $\bar{l}$ gets below min_{th}. ARED attempts to decouple queue length from drop probability to adapt to a range of network scenarios, but can not completely remove instability. In our simulations, we also found it hard to control, or even to predict, the average queue length in steady load.

The developers of ARED also proposed a new AQM algorithm called Blue[4]. Blue neither averages queue length nor uses thresholds. Instead, it drops packets according to a probability independent of queue length, and adjusts the probability in response to buffer overflow (tail-drop) and underflow (empty) events. Blue can be implemented very simply, and requires very little processing power. However, it is difficult to configure for a range of environments.

The PI controller applies control theory to obtain stability[6]. Based on analysis with linearized models of TCP and RED[8], the PI controller is proven to be locally stable. It maintains local stability by guaranteeing gain and phase margins of the system function. However, the stability is restricted by both maximum number of connections and maximum round-trip time. The recommended settings can increase phase margins as link capacity increases, resulting in slow response time.

AVQ takes an approach similar to that of the PI controller to achieve local stability[5]. It maintains a virtual queue with capacity smaller than the actual capacity. On every packet arrival, AVQ updates the virtual capacity and drops the received packet if the virtual queue overflows. Unlike RED, it regulates utilization rather than queue length, and thus has difficulty obtaining a desired queue length and in controlling queuing delay.

Ⅲ. Design and Configuration

The purpose of AQM algorithms is to maintain high link utilization with low loss rate and queuing delay[1],[9]. Stable operation and quick response to changes in load are also desirable. This section outlines the HRED algorithm and explains its design in terms of these attributes. We begin with a general discussion of how AQM algorithms provide control of queuing delay, then provide a simple analysis of local stability, *i.e.*, stability under steady load, showing that HRED improves upon previous algorithms. We finish the section with an analysis of response time.

The first design criteria of AQM is its ability to control queuing delay, which is measured by the expected delay and the expected variance. To provide independent control of both measures, an algorithm requires at least two parameters. The PI controller, for example, has only a single parameter, q_{ref}, for the desired queue length. This parameter allows straightforward control of the average delay, but the variance in delay depends on parameters that offer no intuitive relationship to delay. As delay is proportional to queue length, RED and its variants bound delay via the queue thresholds. HRED uses the same dual-threshold model to control the queuing delay in steady load. As with RED, delay can vary within the operating range.

One of the most difficult problems of AQM is stability. In this paper, we restrict our interests on local stability due to lack of space. Since the network system as a whole ---including the AQM algorithm, cooperative end-hosts, and end-to-end delay--- forms a closed feedback system[8], algorithmic stability can be analyzed through appropriate modeling of each element.

For a bottleneck link shared by N TCP connections with a round-trip propagation delay of RTT, the expected window size of each connection is proportional to $\frac{1}{\sqrt{\overline{p}}}$, where $\overline{p}$ is the expected steady-state drop probability[10]. In steady load, the queue holds the sum of all connections' windows minus the data in flight in the network. Thus, the expected average queue size $\overline{l}$ can be written in terms of $\overline{p}$ as

$$\overline{l} = N\frac{K}{\sqrt{\overline{p}}} - C \times RTT, \tag{1}$$

where K is a constant of proportionality (*i.e.*, $K/\sqrt{\overline{p}}$ is the expected window size)1).

For an AQM algorithm to be locally stable, a deviation from the expected average queue length at time t must not lead to a larger deviation at a later time. More formally, let l_t vary from $\overline{l_t}$ by some small amount δl, with

$$l_t = \overline{l_t} + \delta l. \tag{2}$$

The deviation in average queue length increases the drop probability, which in turn (after a delay) reduces expected window sizes and the expected average queue length. For RED, the impact of the deviation in average queue length is governed by the mapping function

$$p(l) = p_{\max}\frac{l - min_{th}}{max_{th} - min_{th}}. \tag{3}$$

Substituting (2) into (3) at time t, we obtain

$$\begin{aligned} p_t(l_t) &= \overline{p_t} + \frac{p_{\max}}{max_{th} - min_{th}}\delta l \\ &\equiv \overline{p_t} + \delta p. \end{aligned} \tag{4}$$

After some delay τ (several RTTs in practice), the window sizes reflect the change in p. Substituting (4) into (1) at $t+\tau$ and expanding the radical linearly around $\overline{p_t}$ (δp is small), we find

$$\begin{aligned} \overline{l_{t+\tau}} &= \overline{l_t} - \frac{\overline{l_t} + C\times RTT}{2\,\overline{p_t}} \frac{p_{max}}{max_{th} - min_{th}}\delta l \\ &\equiv \overline{l_t} + S(\overline{l_t})\ \delta l. \end{aligned} \tag{5}$$

The coefficient $S(\overline{l_t})$ in (5) denotes the system gain, and indicates how a deviation in RED's average queue length at time t affects the expected average queue length at time $t+\tau$. If $S(\overline{l_t}) < 1$, the system is locally stable. If $S(\overline{l_t}) \geq 1$, the system MAY be unstable. The ambiguity in the latter statement arises because $S(\overline{l_t})$ represents a conservative approximation of system dynamics.

HRED meets the stability criterion by introducing another parameter, p_{min}. It specifies the drop probability when $\overline{l} = min_{th}$. Then, the mapping function (3) becomes

$$p = (p_{max} - p_{min})\frac{\overline{l} - min_{th}}{max_{th} - min_{th}} + p_{min}, \tag{6}$$

and $|S(\overline{l})|$ is given by

1) (1) presents Congestion Avoidance algorithm linearized near an operating point. The linearization limits our proof to local stability.

$$|S(\bar{l})| = \frac{\bar{l}+C\times RTT}{2\bar{p}} \frac{p_{max}-p_{min}}{max_{th}-min_{th}}. \qquad (7)$$

Equation (7) allows the system gain to be controlled through $p_{max}-p_{min}$ to ensure that the system is locally stable. For local stability independent of the current drop probability, HRED maintains $|S(\bar{l})|<1$ by setting p_{max} according to the equation

$$p_{max}-p_{min}=\frac{1}{\chi} \frac{p_{min}}{max_{th}}(max_{th}-min_{th}). \qquad (8)$$

The parameter χ depends on queue parameters and the network capacity.

As well as stability, AQM should have quick response time according to change of traffic loads. Since HRED adjusts p_{min} and p_{max} as shown in Fig. 1, it may exhibit unstable behavior if it is overly aggressive in adjustments to the probability parameters. Conservative adjustments, on the other hand, can lead to lack of responsiveness to change. Stability and response time thus offer tradeoff in terms of the adjustment algorithm parameters, α and β.

Suppose a sudden increase of connections from N to $(1+g)N$. The desired drop probability $\bar{p}'$ to maintain $\bar{q}$ can be given by from (1),

$$\bar{p}'=(1+g)^2\bar{p}. \qquad (9)$$

As an RTT is required before a change of p affects incoming traffic, we configure α and β for HRED to have the drop probability of $(1+g)^2p$ after one RTT.

Considering that the drop probability increases by $(1+g)^2p-p=g(2+g)p$, we set $\alpha=K_\alpha \cdot p(q-max_{th})$, where K_α is a constant, so that HRED has response time independent of current status and change in traffic load. We then estimate the increment on a packet arrival from the algorithm in Fig. 1. Given an increase of N to $(1+g)N$, the total incoming traffic rate increases from $(1-p)C$ to $(1-p)C(1+g)$ and, in turn, the queue length for an RTT increases by

```
if ( $l$ > $max_{th}$ )
    $p_{min}$ = $p_{min}$ + $\alpha$
if ( $l$ > $min_{th}$ )
    $p_{min}$ = $p_{min}$ - $\beta$
$p_{max}$ = $p_{min}$ + $1/\kappa$ $p_{min}/max_{th}$ $(max_{th}$ - $min_{th})$
```

Fig. 1 Adjustment algorithm of HRED.

$(1-p)gC\times RTT$, which corresponds to[2] $(q-max_{th})$. As drop probability is updated on every packet arrival, the increment of drop probability must satisfy

$$\frac{K_\alpha p(1-p)gC\times RTT 1+g)C\times RTT}{S(1-p)} \qquad (10)$$

where S is the average packet size, and the fractional factor in the equation gives the average number of arrivals during an RTT, and the right hand side $g(2+g)p$ is the targeted increment of drop probability during an RTT, *i.e.*, $\bar{p}'-\bar{p}=(1+g)^2p-p$. From (10), we then obtain

$$K_\alpha=\frac{S}{(C\times RTT)^2} \frac{2+g}{1+g}, \qquad (11)$$

which can be approximated by $\frac{2S}{(C\times RTT)^2}$ for $g \ll 1$.

Similarly, β can be replaced with $K_\beta \times p(min_{th}-q)$ and $K_\beta \approx \frac{2S}{(C\times RTT)^2}$.

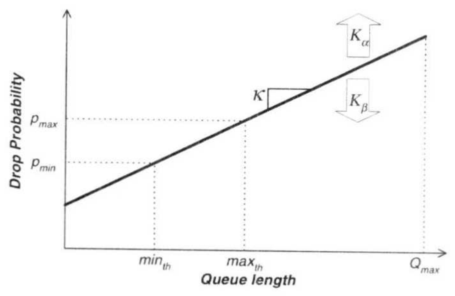


Fig. 2 Drop Probability of HRED.

2) We assume a **sudden** increase in the traffic rate. If a gradual increase, such as that of the TCP fluid flow model, is assumed, the added queue length should be halved.

The parameters K_α and K_β need not be equal. As response time is more of a concern for decreases in traffic load than is stability, K_β can be larger to improve response time. For example, with $Q_{max} = 2\ max_{th} = 4\ min_{th}$, K_β can be twice as much as K_α, as $Q_{max} - max_{th} = 2\ min_{th}$.

Additionally, HRED removes EWMA of RED and extends linear range in mapping function for stability and quick response in changes of traffic load. Since these are related to global stability, we omit detailed explanation. Fig. 2 shows the modified mapping function of HRED.

Ⅳ. Simulation Comparison

This section uses the ns-2 simulator[11] to compare HRED with RED, PI, and AVQ. We include neither ARED nor Blue because they do not try to keep constant queue length.

We first explore queuing delay by measuring queue length and standard deviation in steady load. The results corroborate the analytic claims of the previous section, demonstrating that RED fails to impose adequate controls on queuing delay. We next compare the dynamic behavior of RED, HRED and PI through an investigation of response time to changes in load, *i.e.*, the time required to reach steady state after the traffic load changes. We find that HRED provides faster and more predictable response times than PI with moderate and heavy traffic loads. AVQ is not included in this comparison due to difficulty in measuring response time. The difficulty comes from its characteristic trying to keep queue length as small as possible and lack of control of queue length at a fixed level.

Except where otherwise specified, the simulations are based on a dumbbell topology in which all connections traverse a single bottleneck link with capacity 32 Mbps. The bottleneck link queue operates in byte mode. We assign a random round-trip time between 160 and 240 ms to each connection (uniform distribution), which includes everything except queuing delay at the bottleneck. One set of random RTT's is used in all experiments. Packets on all connections average 500 bytes in length. AQMs are configured as recommended[2],[5],[12] with 500 KB queue size. The configurations in simulation are shown in Table 1 and 2.

Table 1. Configuration of AQM for Simulation

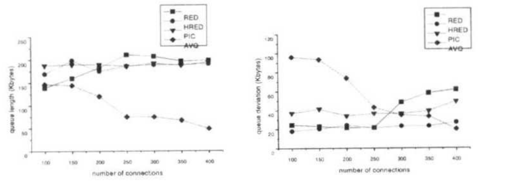
RED	min_{th}=125KB, max_{th}=250KB, w_q=0.002
HRED	min_{th}=125KB, max_{th}=250KB, w_q=1
PIC	q_{ref}=187KB, Sampling 160Hz

Table 2. Parameter Values of AQM for Simluation

RED	p_{max}=0.02, **gentle_** option on
HRED	K_α=1.3563x10^{-10}, K_β=2.7126x10^{-10}, κ=2
PIC	a=1.1379x10-8, b=1.1349x10-8
AVQ	α=0.15, γ=1.0

We monitored average queue length and standard deviation over a period of 200 seconds. The load consisted of a combination of greedy FTP connections and HTTP sessions, with 100-1000 FTPs and 300-3000 HTTPs for each measurement. The results in different traffic loads appear in Fig. 3 and 4. The horizontal axis in each graph reports the number of FTP connections; the number of HTTP connections is proportional.

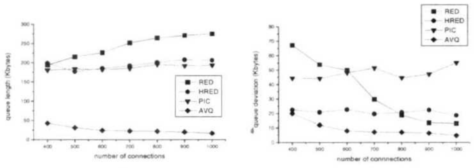
The experiments demonstrate that only HRED manages to provide stable, predictable queuing delay and full utilization of the bottleneck link. RED, PI, and AVQ fail in some network environments through oscillatory behavior, which is observed by large deviation over 30 in Fig. 3



(a) Average Queue Length (b) Standard Deviation

Fig. 3 AQM algorithms in steady state with moderate traffic loads.

(b) and 4 (b).



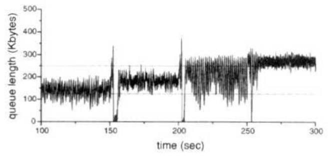
(a) Average Queue Length (b) Standard Deviation

Fig. 4 AQM algorithms in steady state with heavy traffic loads.

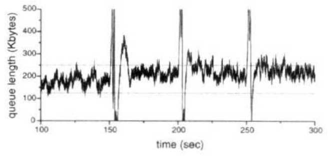
For behaviors of AQMs in dynamic traffic loads, we observe queue length of them. We begin with 100 FTP connections and 300 HTTP sessions, doubled every 50 second after initial 150 second. The results are shown in Fig. 5. The two horizontal lines represent min_{th} and max_{th} for RED and HRED. They also appear in PI and AVQ for comparison purpose. RED loses stability in some ranges of network environment, especially between 200 and 250 second. HRED has quick response time while maintaining stability in all traffic changes. PI gets slower and fails to respond quickly in heavy traffic loads after 250 second. AVQ has an interesting property of keeping the queue length as small as possible without hurting the utilization. Though it performs well in heavy traffic loads, it shows oscillatory behavior in light traffic loads because it does not consider queue deviation. Hence, RED, PI, and AVQ do not meet our criteria of stability and fast response time for dynamic traffic load.

We next compare HRED with RED and PI focusing on response time to changes of traffic load. We measure the time when the queue length gets between $\frac{1}{2}\,min_{th}$ and $\frac{3}{2}\,max_{th}$ for 10 seconds after traffic load changes. Due to difficulty in measuring, we leave out a comparison with AVQ. The basic simulation environment is the same.

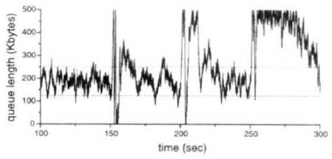
We explored the differences of three algorithms for several network environments as before. In the experiments, the initial traffic of moderate loads



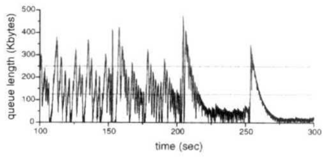
(a) RED



(b) HRED



(c) PIC



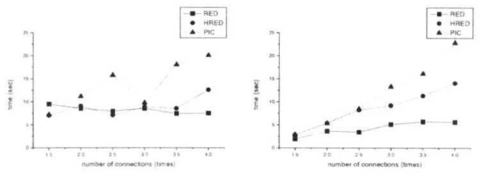
(d) AVQ

Fig. 5 Queue length of AQM schemes in dynamic traffic loads.

started with 100 FTP connections and 300 HTTP sessions and increased by a factor of x, which is horizontal axis. In case of heavy loads, we started simulation with 400 FTP connections and 1200 HTTP sessions. We report two response times: the time for traffic increasing from the initial load, and the time for traffic returning from the increased load to the initial load.

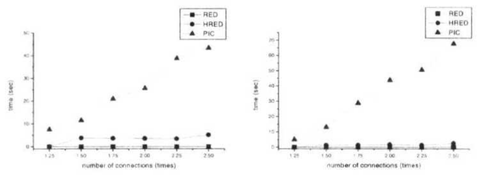
The results of the experiment demonstrate that HRED provides good responsivity to changes in traffic load in Fig. 6 and 7. The PI controller is much slower than HRED to respond to load variations especially in heavy traffic loads.

Since it is essential for a successful AQM algorithms to achieve both stability and fast response time in any traffic loads, HRED is the



(a) Increase (b) Decrease

Fig. 6 Response time in moderate traffic loads.



(a) Increase (b) Decrease

Fig. 7 Response time in heavy traffic loads.

only alternative that can replace the traditional tail-drop mechanism in the Internet, the traffic characteristics of which is very dynamic.

V. Conclusions

We have presented the design and analysis of Hybrid RED, an active queue management algorithm that blends aspects of recent AQM efforts into a RED core, providing the best of both. HRED employs a linear mapping from instantaneous queue length to drop probability across all values of queue length, decoupling queue length management from adaptation to dynamic load. HRED handles changes in load with a simple control algorithm that manages the slope of the drop probability mapping function. It also provides simple means of configuration for achieving stable operation under both steady and dynamic traffic loads.

The main advantages of HRED are its stable operation and fast response time in a wide range of network environments. While RED and AVQ lose stability in moderate and light load, and PI has very slow response time especially in heavy load, HRED makes stable operations without sacrificing response time. The advantages are ensured by simulations in various steady and dynamic traffic loads. For future work, we plan to implement and evaluate HRED on a real network.

References

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

[2] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance", *IEEE/ACM Transactions on Networking,* vol. 1, no. 4, August 1993.

[3] W. Feng, D. Kandlur, D. Saha, and K. Shin, "A Self-Configuring Gateway", *INFOCOM 99*, March 1999.

[4] W. Feng, D. Kandlur, D. Saha, and K. Shin, "Blue: An Alternative Approach To Active Queue Management Algorithms", *NOSSDAV 2001*, June 2001.

[5] S. Kunniyur and R. Srikant, "Analysis and Design of an Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management", *SIGCOMM 2001*, August 2001.

[6] C. Hollot, V. Misra, D. Towsley, and W. Gong, "On Designing Improved Controllers for AQM Routers Supporting TCP Flows", *INFOCOM 2001*, April 2001.

[7] S. Floyd, "Recommendation on using the gentle_ variant of RED", *URL http://www.aciri.org/floyd/red/gentle.html*, March 2000.

[8] C. Hollot, V. Misra, D. Towsley, and W. Gong, "A Control Theoretic Analysis of RED", *INFOCOM 2001*, April 2001.

[9] S. Athuraliya, V. Li, S. Low, and Q. Yin, "REM: Active Queue Management", *IEEE Network*, May 2001.

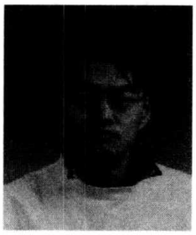
[10] J. Mahdavi and S. Floyd, "TCP-Friendly Unicast Rate-Based Flow Control", *Technical note sent to the end2end-interest mailing list*, January 1997.

[11] The UCB/LBNL/VINT Network Simulator (NS), *URL http://www.isi.edu/nsnam/ns.*

[12] S. Floyd, "RED: Discussions of Setting

Parameters", *URL http://www.aciri.org/floyd/REDparameters.txt*, November 1997.

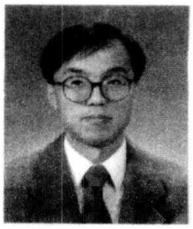
주 창 희(Changhee Joo)



1998년 2월 : 서울대학교 전기공학부 졸업
2000년 2월 : 서울대학교 전기공학부 석사
2000년 3월∼현재 : 서울대학교 전기공학부 박사과정

<주관심 분야> 컴퓨터 네트워크

박 세 웅(Saewoong Bahk) 정회원



1984년 2월 : 서울대학교 전기공학과 졸업
1988년 2월 : 서울대학교 전기공학과 석사
1991년 : University of Pennsylvania 박사

<주관심 분야> 컴퓨터 네트워크