Dynamic threshold control of RED for establishing fairness among thousands of TCP connections

Go Hasegawa and Masayuki Murata
Cybermedia Center, Osaka University
1-30, Machikaneyama, Toyonaka, Osaka 560-0043, Japan
E-mail: {hasegawa, murata}@cmc.osaka-u.ac.jp

Abstract

RED (Random Early Detection) routers can provide better fairness among many competing TCP connections than TD (Tail-Drop) routers, but it is true only when the control parameters of RED are set appropriately according to the network congestion status. However, it is very difficult because the network condition varies largely, especially when the number of accommodated connections is large. In this paper, therefore, we propose dt-RED (RED with dynamic threshold control), which dynamically regulates the RED parameters according to the observed behavior of the RED queue. We confirm the effectiveness of the proposed algorithm through some simulation experiments, and the results show that it can provide better fairness than the original RED, and TD routers, without careful setting of the parameters.

1 Introduction

Many research efforts have been devoted to avoiding and resolving the network congestion. One of them is to reveal the behavior of TCP (Transmission Control Protocol) [1] widely used in the current Internet, and many improved algorithms of the congestion control algorithm of TCP have been proposed in the past literature.

Another approach to deal with the Internet congestion is to deploy the Active Queue Management (AQM) algorithms to Internet routers for providing high performance and fairness among competing connections. One of such algorithm is RED (Random Early Detection) [2], which drops incoming packets at a certain probability when it detects an incipient network congestion, while the traditional TD (Tail-Drop) routers simply discards all arriving packets when the router buffer is fully occupied. While the original idea of the RED algorithm is to avoid consecutive dropping of packets belonging to the same connection, it also has a capability of providing better fairness among connections than TD routers by spreading packet losses. When the control parameters are chosen appropriately, RED routers can provide further better fairness among many TCP connections and it can keep as high throughput as TD routers.

The problem is the difficulty of the parameter setting of RED, that is, we could not find one parameter set to make the RED algorithm work effectively in various network conditions. If RED router is to be effective, its control parameters must be set to values appropriate to the condition of the network (link bandwidth, the number of active connections, congestion level, and so on). As pointed out by other researchers, however, it is very difficult to set them to cope with the dynamically changing network condition [3, 4]. If the parameters are misconfigured its throughput is sometimes lower than that of TD routers. In [3, 4], the authors have thus concluded that there is few reasons to deploy the RED algorithm to Internet routers due to the difficulty of the parameter setting. To solve this problem, several algorithms that dynamically change the control parameters according to queue length, the number of active connections, and so on have therefore been proposed [5-7]. They focus on the dynamic control of the packet discarding probability p, but a rapid change of p degrades the performance of TCP connections, since the throughput of the TCP connection is directly affected by the packet loss ratio [8].

In this paper, therefore, we describe another idea to solve the problem. We propose *dt-RED*, by which we improved the RED algorithm by adding a simple mechanism that sets the threshold values dynamically. We evaluate the effectiveness of this improved algorithm through simulation by comparing it with TD and RED routers, and show that it can provide good fairness in various network congestion states by using one parameter set.

The rest of this paper is organized as follows. Section 2 briefly explains the TD and RED disciplines and their characteristics. Section 3 describes our improved RED algorithm, called dt-RED, and shows some simulation results demonstrating its effectiveness in Section 4. Section 5 concludes the paper with a brief summary and some future research topics.

2 TD and RED Routers

Historically, Internet routers have used a TD (Tail Drop) discipline as a buffer management mechanism: the TD router serves incoming packets in order of their arrival and simply discards newly arriving packets when the buffer is full. The problem with this mechanism is that routers tend to discard packets in bursts [9], which results in packets from the same connection being likely to be discarded. As a result, the fast retransmit algorithm does not help avoid timeout expirations, and this leads to the global synchro-

nization problem [10]. As a result, the TD router cannot provide high throughput and fairness among competing TCP connections [2].

The RED (Random Early Detection) algorithm [2] is designed to cooperate with the congestion control mechanism of the TCP. It detects the beginning of congestion by monitoring the average queue size at the router (the average number of packets in the router buffer) and notifies TCP senders that congestion has occurred by intentionally dropping packets at a certain probability. The RED algorithm sets the packet dropping probability as a function of the average queue size. By keeping the average queue size low, burst packet dropping can be avoided even when packets from the same connection arrive continuously. That is, the algorithm has no bias against bursty traffic. More specifically, it uses a low–pass filter with an exponentially weighted moving average when calculating the average queue size avg, which is maintained and compared with two thresholds: a minimum threshold (min_{th}) and a maximum threshold (max_{th}). The packet dropping probability is determined in different ways according to the queue size avg:

- If $avg < min_{th}$, all arriving packets are accepted.
- If $min_{th} < avg < max_{th}$, arriving packets are dropped with probability $p_{\text{red}}(avg)$, which is defined as follows:

$$p_{\text{red}}(avg) = \frac{avg - max_{th}}{max_{th} - min_{th}} max_p \qquad (1)$$

- If $max_{th} < avg$, all arriving packets are dropped.

The RED router helps prevent TCP's retransmission timeouts, and most lost packets are thus retransmitted by the fast retransmit algorithm. It also helps avoid the phase effect [9] causing all connections to exhibit the similar window changes.

The problem with RED is the difficulty of the parameter setting of RED. In recent works [3, 4] the authors have pointed out that it is difficult to choose the control parameters of RED ($max_{th}, min_{th}, max_p$) to work well, and even when those are appropriately configured, the RED routers cannot provide good performance compared with the TD routers. As opposed to these results, we present a new observation in this paper that RED is still useful, especially from a viewpoint of the fairness among many TCP connections.

3 dt-RED

3.1 Motivation

A RED router is fairer than a TD router, but it can provide better fairness only when the control parameters, especially the threshold values (max_{th} and min_{th}), are set appropriately [3, 4]. The problem is that appropriate values of the control parameters changes dependently on the condition of the network (link bandwidth, the number of active connections, congestion level, and so on). As pointed out by other researchers, it is very difficult to set them to cope with the dynamically changing network condition [3, 4].

To solve this problem, several algorithms that dynamically change the control parameters according to queue length, the number of active connections, and so on have been proposed [5-7]. They focus on the dynamic control of the packet discarding probability of RED p, but a rapid change of p degrades the performance of TCP connections, since the throughput of the TCP connection is directly affected by the packet loss ratio [8]. More important, the previous algorithms do not consider the case where many TCP connections share the bottleneck link. The level of congestion at a bottleneck router that has more than 1,000 connections active at the same time varies widely because each connection reacts to the network congestion independently. Therefore, when the above algorithms are applied, p is likely to fluctuate over a wide range, making the router unstable. Those algorithms thus do not resolve the problems pointed out in [3, 4], and p should be changed as smoothly as possible.

We therefore describe an improved algorithm we call *dt-RED*, which sets the threshold values dynamically. It is very simple compared with the existing approaches but works well, especially with regard to fairness among connections. It does not change p directly, but instead controls the threshold values to change p more smoothly.

3.2 Algorithm

To set the threshold values (max_{th} and min_{th}) appropriately according to the network condition, we developed an algorithm that monitors avg and controls the threshold values dynamically according to the change of avg, so that avg is always between max_{th} and min_{th}. The threshold values are tuned at a regular interval, which in this paper is called an "observation interval." The router monitors avg during every observation interval and sets the threshold values at the end of the interval.

- **Control of max_{th}**

  When avg reaches $M_{max} \cdot max_{th}$ more than once during the observation interval, max_{th} is updated as follows:

$$max_{th} \leftarrow \min(max_{th} + \alpha_{max} \cdot B, M_B \cdot B) \qquad (2)$$

  where M_{max} is a margin ratio for avoiding avg from reaching max_{th}, and M_B is a control parameter that determines the maximum values of max_{th}.

  Otherwise, when avg is enough low and satisfies

$$avg - min_{th} < M_{max} \cdot (max_{th} - min_{th}), \qquad (3)$$

  max_{th} is decreased as follows:

$$max_{th} \leftarrow \max(max_{th} - \beta_{max} \cdot (max_{th} - min_{th}), min_{th}) \qquad (4)$$

  Note that α_{max} and β_{max} respectively determine the increase and decrease ratios of max_{th}.

- **Control of min_{th}**

  When avg decreases to less than min_{th} during the observation interval, dt-RED changes min_{th} as follows:

$$min_{th} \leftarrow \max(min_{th} + \alpha_{min} \cdot (max_{th} - min_{th}), max_{th}) \quad (5)$$

  Conversely, when avg is always larger than min_{th}, min_{th} is decreased in order to shorten the queue length at the router buffer:

$$min_{th} \leftarrow \max(min_{th} - \beta_{min} \cdot (max_{th} - min_{th}), 0) \quad (6)$$

  where α_{min} and β_{min} are respectively the control parameters determining the increase and decrease ratios of min_{th}.

The key issue in dt-RED is how to set M_{max}. That is, when is max_{th} increased or decreased according to the change of avg? It should be set appropriately by considering the fluctuation of avg because the change of avg is greatly affected by the number of active TCP connections passing through the dt-RED router. Therefore, the value of M_{max} should be related to the observation interval. Suppose that the observation interval is set to the time during which the router receives N_p packets, where N_p is a fixed value. Then, M_{max} should also be fixed at a certain value since the number of packets arriving at the router during the observation interval is constant. When the observation interval is fixed, on the other hand, M_{max} should become proportional to the number of connections since the number of arriving packets is proportional to the number of connections. For simplicity, we set the observation interval to be the time during which 100 packets arrive at the router, and we also set M_{max} to 0.9 in the following simulation results. When we want to use the fixed duration of the observation interval and set M_{max} proportional to the number of connections, we have to estimate the number of active TCP connections at the router. One way to do that might be by using a *Zombie List* [11], where the number of active connections can be estimated in $O(1)$ complexity, but we need further research to assess the applicability of the Zombie List.

Dt-RED uses six new control parameters to tune the two parameters of RED, max_{th} and min_{th} dynamically. Dt-RED differs from RED, however, in that the parameter setting of dt-RED has little effect on the fairness of dt-RED. We use only *one* parameter set to deal with various network configurations in dt-RED.

4 Evaluation

In this section, we show several simulation results demonstrating the fairness of dt-RED.

4.1 Simulation Model

Fig. 1 depicts a network model used in the following simulation. It consists of sender hosts, a receiver host, a router, and links interconnecting the hosts and the router.

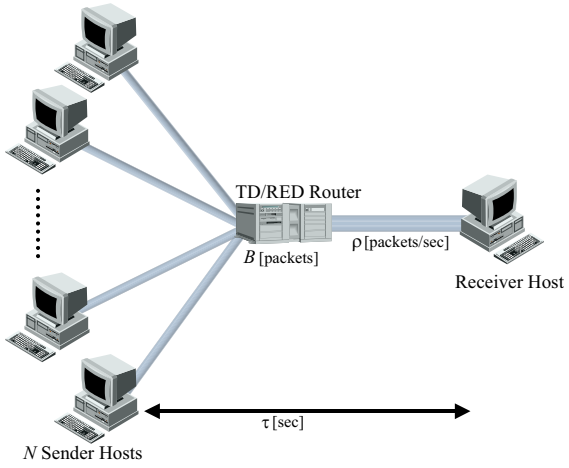


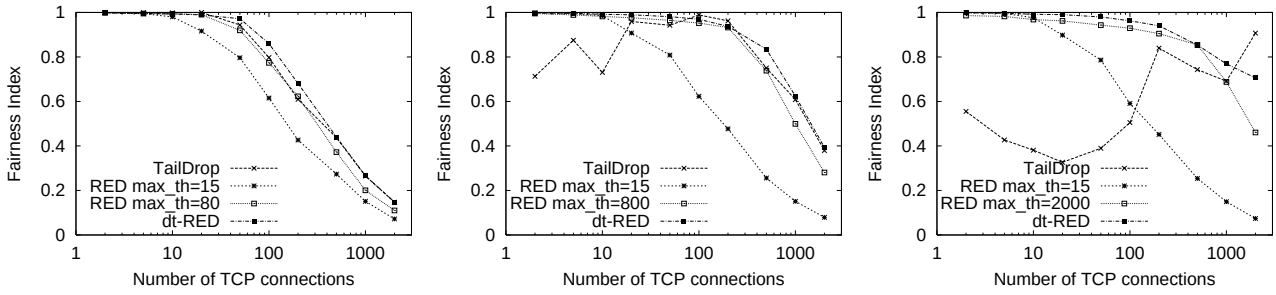
Figure 1: Network model.

N sender hosts share a bottleneck link of ρ [packets/sec], and sends data packets to the receiver host by TCP Reno. The propagation delay between the sender hosts and the receiver host is τ [msec]. The intermediate router has a buffer of the TD or RED discipline. The buffer size is represented by B [packets]. We set the bandwidth and the propagation delay of the link between the sender hosts and the router to 100 [Mbps] and 2 [msec]. We compare three kinds of buffering disciplines at the bottleneck router; Tail-Drop (TD), RED, and dt-RED. For RED, we use $min_{th} = 5$ and $max_p = 0.1$, $w_q = 0.002$, which are the values recommended in [2], and we vary max_{th} to find the best setting of RED. For dt-RED, we use $N_p = 100$ [packets], $M_{max} = 0.9$, $M_B = 0.8$, $\alpha_{max} = \alpha_{min} = \beta_{max} = \beta_{min} = 0.05$.

In the simulations, each sender host sends an infinite size of the document by FTP (File Transfer Protocol). Note that we have also examined the case where each sender host transfers WWW documents, the size distribution of which follows the one shown in [12]. Due to space limitation, we omit the results, but we confirmed the same results as shown in the below. Also note that we have evaluated dt-RED when we use the fixed duration of an observation interval and set M_{max} proportional to the number of connections, and we have also obtained the same results as what follows.

4.2 Results and Discussions

We first show the dynamical change of the threshold values of dt-RED, according to the change of the network congestion status. We set $\rho = 1.5$ [Mbps], $\tau = 4$ [msec], and $B = 10,000$ [packets], and change the number of TCP connections, N, as follows; $N = 500$ from 0 [sec] to 200 [sec] of the simulation time, $N = 1,000$ from 200 [sec] to 300 [sec], $N = 500$ from 300 [sec] to 400 [sec], and $N = 10$ from 400 [sec] to 500 [sec]. The changes of the instantaneous queue length, the average queue length (avg) and max_{th} are shown in Fig. 2 as



(a) B = 100 [packets] (b) B = 1,000 [packets] (c) B = 10,000 [packets]

Figure 3: Fairness evaluation result for ρ = 1.5 [Mbps] and τ = 4 [msec].

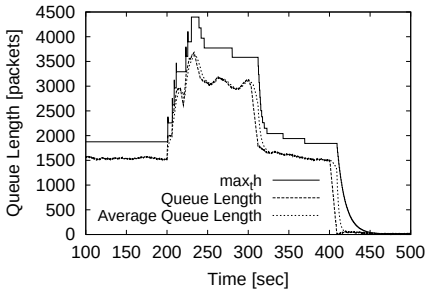


Figure 2: Dynamical change of max_{th} in dt-RED.

functions of the simulation time. We can see that dt-RED changes max_{th} as the queue length at the router buffer increases and decreases.

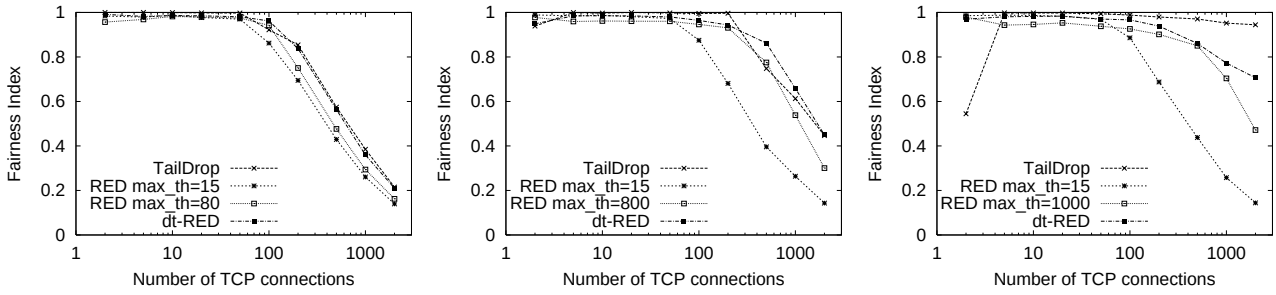
We next show the fairness aspect of the three mechanisms (TD, RED and dt-RED). We first set $\rho = 1.5$ [Mbps] and $\tau = 4$ [msec], and set B to 100, 1,000, and 10,000 [packets]. The results are shown in Fig. 3, where the fairness index values of the three mechanisms are plotted against the number of TCP connections. For RED, we show two cases: one with $max_{th} = 15$ (the value recommended in [2]), and the other with the best value that we found through trial-and-error. From these results, we can make several observations.

First, the TD discipline cannot provide fairness among connections at all, especially when the number of connections is large, because the TD discipline has no mechanism for fairness enhancement. Furthermore, the fairness of TD is also degraded when the buffer size is too large compared with the number of connections (Fig. 3(c)). This shows another problem of the TD discipline: when the buffer size per each TCP connection is too large, the buffer occupation becomes different among connections, which leads to differences of the throughput values of TCP connections. That is, larger buffers at TD routers do not always assure fairness among connections.

For the RED discipline, we can see that the recommended values of the control parameters cannot provide the good fairness. In the worst case, RED shows a lower fairness than TD does. The main reason is that the max_{th} in this case is too small for the number of connections and the buffer size. When max_{th} is set appropriately, on the other hand, RED provides better fairness than TD in all cases. That is, RED can work better than TD if the control parameters are properly configured according to the change of the network congestion status. Of course, it is very difficult to find the best value of max_{th} (max_{th} = 80 for Fig. 3(a), 800 for Fig. 3(b), and 2,000 for Fig. 3(c)). An appropriate value cannot be found without many trials.

The fairness index values of dt-RED are almost same as the ones obtained by RED with the best parameter setting. This is a very important characteristics of dt-RED because dt-RED can provide this fairness by using a *single parameter set*; we need not to tune the control parameters of dt-RED according to the change of the network congestion status.

We next change the propagation delay of the bottleneck link, τ, from 4 [msec] to 400 [msec]. The simulation results are shown in Fig. 4, where we can again see that the RED algorithm with the control parameter values recommended in [2] cannot provide the fairness that the TD router can, especially when the number of connections is large. The fundamental reason is that the recommended values are not based on a situation in which there are more than 100 connections at the bottleneck router. The appropriate control parameters set for a given number of connections could of course be found, but only by trial-end error. dt-RED, on the other hand, shows very good fairness regardless of the number of connections and the size of the router buffer. It provides fairness almost same or even better than that provided by RED with the best parameter set. One problem is that when $B = 10,000$ [packets] (Fig. 4(c)) and the number of connections is large, TD is the fairest of the three disciplines. In this case, TD results in no packet loss at the buffer. Since RED and dt-RED routers intentionally discard incoming packets when only a few packets are stored at the router buffer, they are not as fair as a TD router. For further fairness improvement of dt-RED, we should treat



(a) B = 100 [packets] (b) B = 1,000 [packets] (c) B = 10,000 [packets]

Figure 4: Fairness evaluation result for ρ = 1.5 [Mbps] and τ = 400 [msec].

this case more effectively, which is left to be a future work.

5 Conclusion

This paper described the improved algorithm of RED we call dt-RED, which changes the RED's threshold values (max_{th} and min_{th}) dynamically according to the network congestion status. Dt-RED adds a very simple mechanism to the original RED but it can provide greatly improved fairness among many TCP connections without any parameter tunings. We have shown its effectiveness through simulation results by comparing it with TD and RED routers.

In this paper the homogeneous network model, where all TCP connections have the same propagation delays, has been used. The fundamental characteristics of dt-RED has been revealed, but in the future we will investigate the performance of dt-RED under the network model in which each TCP connection has a different propagation delay and bandwidth.

Acknowledgements

This work was partly supported by the Research for the Future Program of the Japan Society for the Promotion of Science under the Project "Integrated Network Architecture for Advanced Multimedia Application Systems," and by a Grant-in-Aid for Scientific Research (B) 13450158 from the Japanese Ministry of Education, Culture, Sports and Science and Technology of Japan.

References

[1] W. R. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*. Reading, Massachusetts: Addison-Wesley, 1994.

[2] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, pp. 397–413, Aug. 1993.

[3] Martin May, Jean Bolot, Christophe Diot, and Bryan Lyles, "Reasons not to deploy RED," in *Proceedings of IWQoS'99*, June 1999.

[4] Mikkel Christiansen, Kevin Jeffay, David Ott, F. Donelson Smith, "Tuning RED for web traffic," in *Proceedings of ACM SIGCOMM 2000*, August 2000.

[5] Haining Wang and Kang G. Shin, "Refined design of random early detection gateways," in *Proceedings of Globecom'99*, pp. 769–775, December 1999.

[6] Wu-chang Feng and Dilip D.Kandlur and Debanjan Saha and Kang G. Shin, "A self-configuring RED gateway," in *Proceedings of IEEE INFOCOM'99*, March 1999.

[7] Wu-chang Feng and Dilip D.Kandlur and Debanjan Saha and Kang G. Shin, "BLUE: A new class of active queue management algorithms," Tech. Rep. CSE-TR-387-99, U. Michigan, March 1999.

[8] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: a simple model and its empirical validation," in *Proceedings of ACM SIGCOMM'98*, pp. 303–314, Aug. 1998.

[9] S. Floyd and V. Jacobson, "On traffic phase effects in packet-switched gateways," *Internetworking: Research and Experience*, vol. 3, pp. 397–413, Aug. 1992.

[10] L. Zhang and D. D. Clark, "Oscillating behavior of network traffic: A case study simulation," *Internetwoking: Research and Experience*, vol. 1, pp. 101–112, 1990.

[11] T. J. Ott, T. V. Lakshman, and L. Wong, "SRED: Stabilized RED," in *Proceedings of IEEE INFOCOM'99*, Mar. 1999.

[12] P. Barford and M. Crovella, "Generating representative Web workloads for network and server performance evaluation," in *Proceedings of ACM SIGMETRICS '98*, 1998.