Adaptive Nonlinear RED Algorithm for TCP Congestion Control

Kyung-Joon Park, ∘Eun-Chan Park, Hyuk Lim, and Chong-Ho Choi∗

∗ School of Electrical Engineering & Computer Science, Seoul National University, Seoul 151-742, Korea
(Tel: +82-2-880-7310; Fax: +82-2-874-4035; Email:chchoi@csl.snu.ac.kr)

Abstract: Congestion control is a critical issue in TCP networks. Recently, active queue management (AQM) was proposed for congestion control at routers. The random early detection (RED) algorithm is widely known in the AQM algorithms. We present an adaptive nonlinear RED (NRED) algorithm, which has nonlinear drop probability profile. The proposed algorithm enhances the performance of the RED algorithm by the self-parameterization based on the traffic load. Furthermore, the proposed algorithm can effectively adapt itself between the RED and the drop-tail queue management by adopting proper nonlinearity in the drop probability profile. Through simulation, we show the effectiveness of the proposed algorithm comparing with the drop-tail and the original RED algorithm.

Keywords: random early detection, active queue management, TCP congestion control

1. Introduction

A considerable amount of research has been carried out on TCP congestion control of the Internet to enhance the performance of the network. Congestion control mechanisms can be divided into two categories. One is the modification of the existing TCP algorithm and the other is active queue management (AQM).

The former is based on the congestion window; the sender keeps a congestion window whose size limits its transmission rate. Many mechanisms called as *slow-start, fast-recovery* and *fast-retransmit* have been proposed to effectively regulate the transmission rate of each connection. These window-based mechanisms, however, have a problem that TCP sender reduces its transmission rate only after detecting packet loss caused by queue overflow.

Another way for congestion control has been proposed as a solution for preventing loss due to queue overflow, which is active queue management (AQM). The goal of AQM is to detect congestion early and convey congestion notification to the senders allowing them to reduce their transmission rates before occurring actual packet loss. One of the most important AQM algorithms is RED [1]. RED detects congestion using an exponentially weighted moving average of the queue size, and intelligently drops or marks packets to control the congestion at router buffers. Many variants of the RED algorithm have been proposed by researchers [2][3][4][5][6][7]. In many congestion scenarios, RED gateways can alleviate the problems found in the other AQM algorithms in that they can prevent global synchronization, reduce packet loss rates, and minimize biases against bursty sources using a simple, low-overhead algorithm. However it still remains an inexact science to get appropriate values of RED parameters that show good performance under different congestion scenarios [8][9]. It is pointed out that the average queueing delay and throughput are sensitive to the traffic load and to parameters. There also exists some argument on the deployment of RED, especially in the case of small buffer size [10].

To alleviate these problems, we introduce nonlinear RED (NRED) algorithm, which has nonlinear drop probability profile together with a self-tuning parameter. By introducing a self-tuning parameter based on the traffic load, the proposed algorithm can adapt itself under different congestion scenarios. At the same time, the proposed algorithm can shift efficiently between the RED and the drop-tail algorithm by adjusting the nonlinearity of the drop probability profile. In the following section, we present the proposed algorithm. In Section 3, we compare the NRED algorithm with the drop-tail and the original RED algorithm via simulation using the ns-2 network simulator [11]. Finally we present the conclusion in Section 4.

This work was supported in part by the Brain Korea 21 Program of the Korea Ministry of Education.

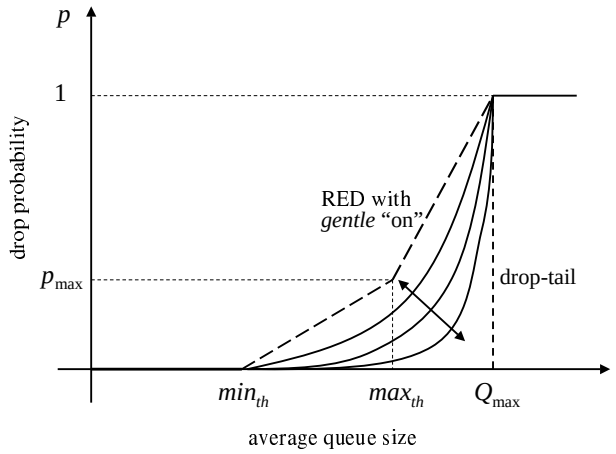

Fig. 1. Drop probability profile of NRED compared with the drop-tail and the RED.

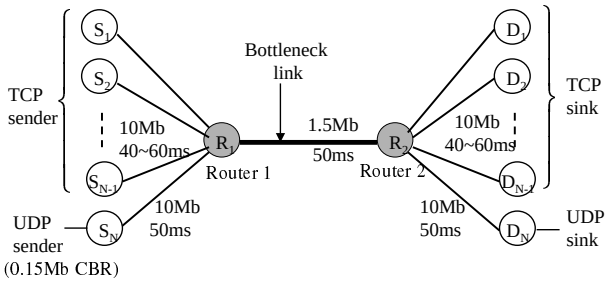

Fig. 2. Network topology for simulations.

2. Proposed algorithm

The NRED algorithm consists of two parts; the nonlinear drop probability profile and the update rule for the profile parameter n.

The drop probability p is calculated as follows:

$$p = \begin{cases} 0 & \text{if } Q_{avg} < min_{th} \\ \left(\frac{Q_{avg} - min_{th}}{Q_{max} - min_{th}}\right)^n & \text{if } min_{th} < Q_{avg} < Q_{max} \end{cases} \quad (1)$$

Here, Q_{max},Q_{avg} and min_{th} denote the buffer size, the average queue size and the minimum threshold, respectively. And n is a self-configuring parameter that updates itself based on the traffic load. The drop probability profile of proposed algorithm is shown in Figure 1, compared with those of the drop-tail and the RED with the *gentle_* option [12].

The adaptation of proposed algorithm can be performed by updating the profile parameter n. The update rule for n is as follows:

```
For every Q_avg update:
  if (min_th < Q_avg < max_th)
    leave n intact
    status = Between;
  else if (Q_avg < min_th && status != Below)
    increase n
    status = Below;
    n = min(n_max, n + 1);
  else if (Q_avg > max_th && status != Above)
    decrease n
    status = Above;
    n = max(n_min, n - 1);
```

Here, min_{th} and max_{th} are the minimum and the maximum threshold to determine the status of traffic load, respectively; they are the same as in the RED algorithm. If traffic load is normal, and the average queue size is between two thresholds, the profile parameter does not change. If traffic load changes to be light or heavy, the profile parameter increases or decreases. In this way, the proposed algorithm can adapt itself between the drop-tail and the RED algorithms. We can see this by comparing the drop probability profile of the NRED with those of the drop-tail and the RED as shown in Figure 1. As load traffic gets lighter, n increases, thus the drop probability profile gets similar to that of the drop-tail algorithm. On the contrary, as load traffic gets heavier, n decreases and the drop probability profile gets similar to that of the RED algorithm.

The update rule for n is the same as the one for p_{max} of ARED [3]. The main difference between NRED and ARED is the nonlinearity of the drop probability profile. By introducing the nonlinear drop profile, NRED can effectively adjust the drop probability between the drop-tail and the RED algorithms.

3. Simulations

To compare the performance of the NRED algorithm with those of the drop-tail and the RED algorithms, a simple bottleneck network configuration has been implemented using the ns-2 simulator. Figure 2 shows the network topology used in our simulations; the network is configured with two routers (R1 and R2) and a number of TCP connections and one UDP connection. Two routers are connected through

Table 1. The parameters of RED

parameter	description	value
min_{th}	threshold when Q_{avg} should exceed before any packets are dropped or marked.	5 packets
max_{th}	threshold when Q_{avg} should exceed before all packets are dropped or marked.	$3 \times min_{th}$
p_{max}	maximum dropping or marking probability on congestion	0.1
w_q	weight for updating Q_{avg}	0.002

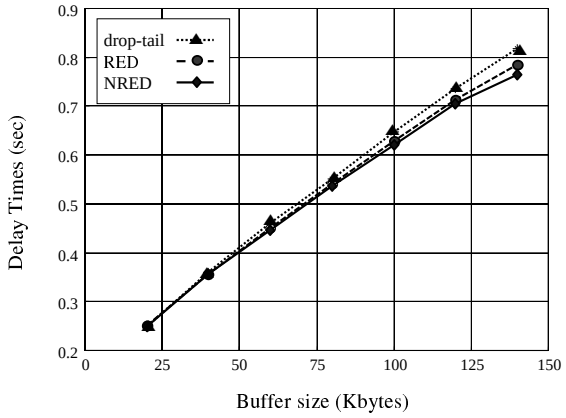

Fig. 3. Simulation results : delay vs. buffer size.

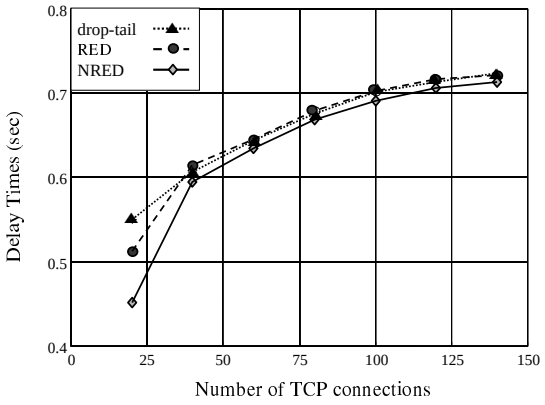

Fig. 5. Simulation results : delay vs. number of TCP connections.

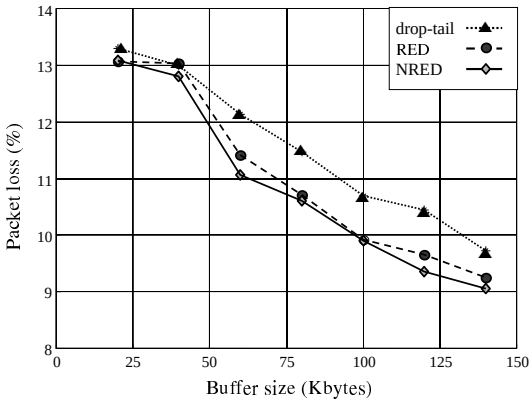

Fig. 4. Simulation results : packet loss vs. buffer size.

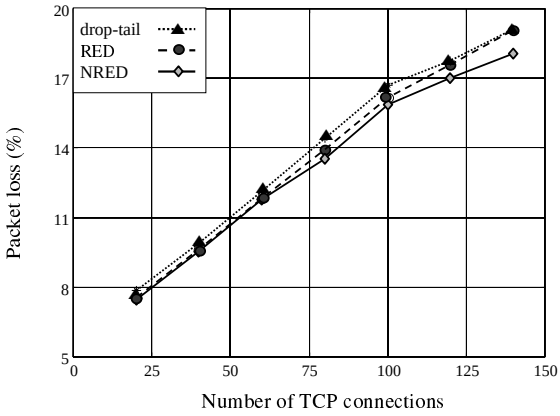

Fig. 6. Simulation results : packet loss vs. number of TCP connections.

a 1.5Mb link when congestion occurs. All other links have the same bandwidth of 10Mb which is sufficient not to create any congestion. Using this network, $N-1$ FTP applications on TCP connections start sending packets at time 0 simultaneously, and a constant-bit-rate (CBR) application on UDP connection persistently sends packet at the rate of 0.15Mb that is one tenth of the bottleneck link capacity. The propagation delays between the sources/sinks and the routers are uniformly distributed between 40ms and 60ms. The propagation delay between the two routers is 50ms. The average packet size is set to 1 Kbyte. The parameters of the RED are listed in Table 1, which are set to the values recommended in [12]. The parameters of the NRED are same those of the RED except that p_{max} is unnecessary.

We compared the performances of the drop-tail, the RED, and the NRED algorithms by varying the buffer size and the number of TCP connections. First, we varied the buffer size while the number or TCP connections was fixed at 50, and then compared the performances of the router. The allocated buffer size of the bottleneck router was changed from 20Kbytes 140Kbytes by 20Kbytes.

The performance of the RED degrades when the buffer size is small [10]. Figure 3 shows the delay of the drop-tail, the RED, and the NRED. When the buffer size is small, all the three algorithms do not show much difference. This can be explained by the fact that, when the buffer size is small, the three algorithms are virtually the same because of the averaging effect of the queue size [1]. However, as the buffer size gets larger, the NRED shows better delay performance among the three algorithms. We also compared the packet loss of each algorithm. The simulation results are shown in Figure 4. These results also show that the performance of the NRED is the best among the three algorithms.

Next, we varied the number of TCP connections while buffer size fixed at 100 Kbytes, and then compared the performances of the router. The number of TCP connections was changed from 20 to 100 by

20. As shown in Figure 5, the delay time of the RED and that of the drop-tail do not show large difference, however, the delay time of the NRED is much smaller than those of the drop-tail and the RED when the number of TCP connections is small. On the other hand, the simulation results of packet loss are shown in Figure 6. The NRED suffers the least packet loss compared with the drop-tail and the RED algorithms. As the number of connections gets larger, the NRED shows better performance in packet loss than the other algorithms.

4. Conclusion

The RED algorithm is simple and effective as an active queue management mechanism. However, there is no single set for the RED parameters that works well under different congestion scenarios. There are also some arguments on the deployment of the RED gateways under some situations such as gateways with small buffer sizes. To alleviate these problems, we have proposed an adaptive RED algorithm with nonlinear drop probability profile. The proposed algorithm can adapt its drop probability based on traffic load, and it can effectively reduce packet loss in congested networks. Simulation results show that the proposed algorithm performs better than the drop-tail and the original RED algorithms in terms of packet loss and delay time.

References

[1] S. Floyd and V. Jacomson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Networking*, vol. 1, no. 4, pp. 397-413, 1993.

[2] W.-J. Kim and B.G. Lee, "FRED-fair random early detection algorithm for TCP over ATM networks," *IEE Electronics Letters*, vol.34, no.2, pp. 152-154, 1998.

[3] W. Feng, D.D. Kandlur, D. Saha, and K.G. Shin, "A self-configuring RED gateway," *Proc. IEEE INFOCOM*, New York, NY, USA, pp. 1320-1328, 1999.

[4] W. Feng, D.D. Kandlur, D. Saha, and K.G. Shin, "Blue: a new class of active queue management algorithms," Tech. Rep., UM CSE-TR-387-99, 1999.

[5] H. Wang, K.G. Shin, "Refined Design of Random Early Detection Gateways," *Proc. IEEE GLOBECOM*, pp. 769-775, 1999.

[6] S. Athuraliya, S.H. Low, V.H. Li, and Q. Yin, "REM: Active queue management," *IEEE Network*, vol.15, pp. 48-53, 2001.

[7] J. Koo et al., "MRED: A New Approach to Random Early Detection," *Proc. International Conf. on Information Networking*, pp. 347-352, 2001.

[8] S.D. Cnodder, O. Elloumi, and K. Pauwels, "RED behavior with different packet sizes," *Proc. ISCC*, pp. 793-799, 2000.

[9] M. Christiansen, K. Jeffay, D. Ott, and F.D. Smith, "Tuning RED for Web Traffic," *Proc. ACM SIGCOMM*, Stockholm, Sweden, pp. 139-150, 2000.

[10] M. May, J. bolot, c. Diot, and B. Lyles, "Reasons not to deploy RED," *Proc. IWQoS*, pp. 260-262, 1999.

[11] "ns-2 network simulator," http://www.isi.edu/ nsnam/ns/.

[12] "RED Queue Management," http://www.aciri.org/ floyd/red.html.