TR-IIS-05-016

Improving End-to-End Performance of RED

Chin-Fu Ku, Ray-I Chang, Jan-Ming Ho, Sao-Jie Chen

September 2005 || Technical Report No. TR-IIS-05-016
http://www.iis.sinica.edu.tw/LIB/TechReport/tr2005/tr05.html

Improving End-to-End Performance of RED

Chin-Fu Ku, Ray-I Chang, Jan-Ming Ho, Sao-Jie Chen

Abstract

Congestion control mechanisms play a important role in today's Internet, since network environments are geting more and more complicated. Besides, new applications request diverse services delivered by the network. Random Early Detection (RED) provides an efficient and low-overhead mechanism to effectively control congestion. Nevertheless, RED would slow down TCP senders' reaction to change of network condition, in particular the bottleneck link with long propagation delay or RED with a smaller weighted factor. Thus, we proposed a better way to drop/mark packets not just based on queue occupancy. In this paper, an enhancement intending to improve the performance in terms of delay, delay jitter and packet loss ratio is proposed. The basis idea is to react quickly to major changes in network conditions so as to shorten the time it takes for a sender to gather signals, e.g., round-trip time and lost packets from networks. The simulation results show the enhancement could effectively raise the performance in terms of delay, delay jitter and packet loss ratio.

1 Introduction

The Internet has been a useful facility for our daily life. As the increasing of applications and users on the Internet, the availability and stability of the Internet has been a major concern to most network operators and researchers. Congestion control is one of the most important mechanism in keeping the Internet, a huge and heterogeneous internetworking, available and stable. Because congestion could highly degrade the service to users, or even make the service unavailable. Many control schemes are proposed, though the causes of congestion are diverse and dynamic. Besides, new network protocols or applications, such as IP telephony or voice over IP, may raise new challenges for congestion control.

A simple definition of congestion could be, ”the sum of demanded resource is greater than available resource”. In a network, resources includes link bandwidth and buffer in intermediate nodes, e.g., switches or routers. However, the causes of congestion varies from situation to situation. In [1], Gevros *et al.* gives an overview on issues of congestion control and related topics.

As network traffic is growing so fast, traditional network protocols are also facing stricter examining and validating. Nowadays, TCP has been the dominating protocol among the Internet traffic. Although TCP got his own long history in controlling network congestion, many improvements and enhancements still are in progress. In [2], Floyd concisely described recent modification and enhancements to TCP's congestion control. Nevertheless, early attentions were put on easier deployments and low complexity. Thus the end to end protocol, i. e. TCP itself, is the focus to control congestion. In fact, they successfully maintain stability of the Internet. But the network condition has become more and more complicated because of the development of more and more advanced Internet technologies. Several alternatives are studied in literatures.

Traditional network intermediate nodes, such as router, switches or bridges, use First-In-First-Out(FIFO) queue to store packets and drop-tail (or called First-Come-First-Serve) discipline to drop incoming packets when the queue is full. However, drop-tail could make large packets loss whenever the queue is full, which might cause TCP synchronization phenomenon. Therefore, some researchers put focus on the design of queueing disciplines on intermediate nodes. Random Early Detection (RED) [3] was proposed by Jacobson and Floyd in 1993 and then many enhancements or variants were proposed.

As it's name suggested, RED prevents buffer overflow by early detecting building up of queue. Once the average queue length exceeds a threshold, it sends signal back to the sender by means of probabilistically dropping or marking, such as Explicit Congestion Notification (ECN) [4], packets. The sender then decreases the sending rate after receiving the signal and thus relieves congestion condition. Furthermore, when RED decides to drops (or marks) packets, it would randomly choose one instead of the last one, thus, it could keep all flows from synchronization phenomenon,

In [5, 6], May *et al.* studied RED's performance based on their theoretical analysis and questioned the deployment of RED. However, their analysis didn't take TCP's control mechanism into account. However, TCP adjusts its sending rate by taking packet drop or explicit notification, such as ECN, as its feedback. Simply modeling input rate pattern as Poisson, Batch Poisson or on-off model does not reflect the reality. RED has been suggested to be deployed in the Internet [7], there are already some products implementing RED or it's variants, e.g., Cisco's Weighted Random Early Detection (WRED).

RED uses exponential weighted moving average (EWMA) to maintain an average queue length (q_{avg}) instead of maintaining instantaneous queue length. For brevity, we also use 'queue length' to denote 'average queue length'. EWMA acts as a low-pass filter to gather long-term trend without influence of short-term fluctuation. If the average queue length is lower than a minimum threshold (min_{th}), RED allows all incoming packets into the queue; if queue length rise over min_{th} but still below maximum threshold (max_{th}), RED uses average queue length to calculate the drop probability and accordingly handles incoming packet with that drop probability. If the queue length is higher than max_{th}, RED would drop any incoming packet unless the parameter, *gentle* mode [8], has been set. If *gentle* mode is on, when average queue length exceeds maximum threshold, another linear function will be used to calculate drop probability

Most active queue management mechanisms adopt moving average to keep queue length. On positive side, it could stabilize the system and provide better indication of unbalance between input and output links. But it also delay the time to detect major network condition changes, especially climbing of input rate. As [9] shown, using smaller w_q would make RED more stable, while smaller w_q would delay the convergence of new value. In addition, when bottleneck link with long propagation delay, the sender would take long time to get the signal of congestion from the network.

If the intermediate node calculate drop probability not just based on averaging queue length, but also the rate of changing of averaging queue length. This should speed up the reaction to network condition changing. In other words, we are interesting in detecting input rate climbing, since that would make queue built up fast. The change rate of queue length is a function of the difference between input output rate. The output rate is determined by output link capacity, so the input rate represents the change of queue length as long as the queue is not empty.

Among many enhancements to RED or variants of RED, Random Early Marking(REM) [10], Adaptive Random Early Detection(ARED) [11] and the new ARED [12] proposed by Floyd *et al.* provide much better improvements and aim to improve delay and packet loss ratio. The method proposed in this paper also aims to reduce delay, delay jitter [1] and packet loss ratio. Since the new ARED are more robust than

[1]We adopt "the variance of delay" as the definition of delay jitter.

original ARED proposed by Feng *et al.*, ARED would refer to the one proposed by Floyd *et al.* in this paper.

This paper is organized as following, Sec 2 would give a brief introduction of background information. Sec 3 describes the method proposed and the Sec 4 shows the simulation results and discussion. At the end, conclusion and remarks would be provided in Sec 5.

2 Background

TCP uses window-based control mechanism to prevent network from congestion. Once a TCP sender recognizes the occurrence of congestion, it will shrink the congestion window size so as to decrease the transmitting rate. The basis of control over congestion window lies in the well-known principle, "Additive Increase, Multiplicative Decrease" (AIMD). In other words, a TCP sender would increase the window size by one segment per round-trip time (RTT) if there is no signal received, otherwise, it would shrink the window size to half of the current size.

TCP detects congestion mainly by the event of packet loss. Some enhancements for TCP use other signal to detect packet loss. For example, fast retransmission would treat receiving three duplicated acknowledgments (ACKs) as the signal of packet loss so as to avoid waiting costly retransmission timeouts. However, these enhancements is hardly applied in slow-start phase. In short, packet loss is an important signal for all TCP senders to react to avoid congestion in networks. And

RED relies solely on the reaction of TCP. However, since packet loss may result in ‘timeout’ if TCP didn’t get enough window size to operate with fast retransmission/recovery. The proposal of Explicit Congestion Notification (ECN) provides a proactive way to signal senders. If the sender, receiver as well as any intermediate nodes along the path are ECN-enabled, the sender might be able to detect congestion without any packet loss. Therefore, the combine of ECN and RED could obtain better performance, especially in packet loss ratio.

In contrast with TCP congestion control mechanism, traditional intermediate nodes were not involved much in congestion control. A FIFO queue are used to store packets when unbalance of input and output rate of a link occurred. Simple drop-tail discipline are used to manage the FIFO queue; accepting the incoming packet and putting into the tail of queue when space is available, otherwise, dropping the incoming packets until there is space. Drop-tail is simple but it could cause so-called TCP synchronization phenomenon in which several TCP senders send packets and backoff simultaneously, eventually degrading the performance. That’s because of the bursty characteristics of TCP traffic, consecutive packets could be dropped once the queue is full.

RED aims to eliminate these drawbacks mentioned above: preventing the queue from full by early detect, preventing TCP senders from synchronous behavior by randomly dropping/marking. For early detection, RED set two thresholds to decide the level of congestion situation.

ARED uses dynamic max_p based on the position of average queue length so as to try to maintain the average queue length in a predefined position. REM use a different way, called price function, to calculate drop/mark probability. Both mechanisms works fine in most situation; their calculation probability only rely on average queue length, so we could expect the inefficiency to link with long propagation delay.

3 Method Proposed

The basic idea behind the proposed method, Rate-Aware Random Early Detection (RARED for short), is as follows: The aggregated input rate could be observed in intermediate nodes, and it could provide more precise information than queue occupancy. RARED detects the change of input rate, in particular, the climbing of input rate, and then handle incoming packets with another drop probability.

As mentioned above, A TCP sender would increase congestion window size if it doesn't receive any signal from network. But the moving average procedure would delay the change of queue length in RED. When RED finds queue occupancy is raising to high level so as to trigger it to drop/mark packets with higher drop probability, most senders has transmitted packets over their deserved share. RARED prevents this situation by observing the change of input rate. By earlier informing senders, intermediate nodes could decrease packet drop rate. In other words, RARED informs senders strongly by dropping packets with higher drop probability at the time of input rate climbing, it can avoid dropping a number of packets later.

Moreover, because RARED earlier informs senders at the time of input rate changing, queue occupancy would be more stable; and therefore delay jitter would be decreased. Keeping queue occupancy small also decrease end-to-end delay, since queuing delay dominates the end-to-end delay in most environments.

We calculate drop probability with two formulas according to two different conditions. We call one condition “input rate climbing” and the other “unchanged.” RARED determines which condition is by comparing changing rate of average queue length to a given threshold $slope_{th}$. Fig. 1 shows RARED algorithm, max_p^r is the maximum drop probability used in “rate climbing” condition. As we evaluated, max_p^r doesn’t affect the performance once it is greater than 0.8.

4 Simulation Results

We implemented the proposed scheme in the *ns-2* network simulator [13] to evaluate the improvement. The network topology is a simple dumbbell as Fig. 2 shown and the queue size of bottleneck link is 30 packets. Two FTP sessions randomly start in between 0 to 0.1 second and last to the end of simulation, i.e. 40 seconds. In the middle of simulation another *m* FTP sessions would randomly start (in between 20 and 20.1 seconds) and last to the end, which is used to simulate the change of network condition. The range of *m* is from 1 to 15. There are one FTP session run over entire simulation to simulate reverse path traffic, which might cause ACK packets loss as well as ACK compression. Besides, there are 20 Web sessions running from begin to the end of simulation on 10 pairs of client and server.

TCP Reno is used for all simulation, the window size is 24 packets and with ECN enabled. The mean packet size is 1000 for both TCP and all queue management methods. The queue management methods evaluated includes RED, REM, ARED and RARED and all with ECN turned on. Parameters used for RED is $max_{th} = 15$, $min_{th} = 5$, $max_p = 0.1$, $w_q = 0.0019$ and with gentle mode on [14]. For REM, $\phi = 1.001$, $\gamma = 0.001$ and $b^* = 20$. We use the default values in *ns-2* for ARED. The slope threshold ($slope_{th}$) and max^r_p for RARED are 20 and 0.8 respectively.

Every simulation result shown in the figure is the average value and 95% confidence interval obtained by performing each simulation ten times.

From Fig. 3, 4 and 5, RARED outperforms other schemes. In other words, by early informing the sender the changing of network condition could prevent the sender from sending too many packets. The queue length could be more steady and less packets would be dropped; therefore, delay, delay jitter and packet loss ratio could be improved. Fig. 6 shows the throughput of all flows, RARED could keep the same throughput as others.

5 Conclusion

We have shown the improvement of the proposed method with simulation under the condition of long link delay. When the network with short link delay , the performance of proposed method is the same as ARED, which we didn't show on this paper due to page limit. In addition to improving performance in terms of delay, delay jitter and packet loss ratio, the method provides network administrators a way to control the service delivered; i.e. tuning the slope threshold. A lower threshold could effectively elevate the performance against to high variation of network condition, but it comes with the price of less link utilization and high possibility of fluctuation.

In the future work, a mechanism to tune *slope threshold* automatically, based on the characteristics of link and the dynamics of network condition, would be investigated. We will also seek the possibility of solving the problem of non-responsible traffic by this methods.

6 Acknowledgment

This work was supported by the National Science Council Grant No. NSC-91-2213-E-001-026.

References

[1] P. Gevros, J. Crowcroft, P. Kirstein, and S. Bhatti, “Congestion control mechanisms and the best effort service model,” *IEEE Network*, vol. 15, no. 3, pp. 16–26, May 2001.

[2] S. Floyd, “A report on recent developments in tcp congestion control,” *IEEE Communications Magazine*, vol. 39, no. 4, pp. 84–90, Apr. 2001.

[3] S. Floyd and V. Jacobson, “Random early detection gateway for congestion avoidance,” *IEEE/ACM Transaction on Networking*, vol. 1, no. 4, pp. 397–413, Aug. 1993.

[4] K. K. Ramakrishnan and S. Floyd, “Proposal to add explicit congestion notification (ECN) to IP,” *IETF Request for Comments, RFC 2481*, Jan. 1999.

[5] M. May, J. Bolot, C. Diot, and B. Lyles, “Reasons not to deploy RED,” in *Seventh International Workshop on Quality of Service (IWQoS’99)*, Jun. 1999, pp. 260–262.

[6] M. May, T. Bonald, and J. Bolot, “Analytic evaluation of RED performance,” in *Proceedings of the Conference on Computer Communications (IEEE INFOCOM)*, vol. 3, Mar. 2000, pp. 1415–1424.

[7] B. Braden, D. Clark, and et al, “Recommendations on queue management and congestion avoidance in the internet,” *IETF Request for Comments, RFC 2039*, Apr. 1998.

[8] S. Floyd, “Recommandation on using the “gentle_” variant of RED,” *[Online] Available http://www.icir.org/floyd/red/gentle.html*, Mar. 2000.

[9] W. Wu, Y. Ren, and X. Shan, “Stability analysis on active queue management algorithms in routers,” in *Ninth International Symposium on Modeling, Analy-*

sis and Simulation of Computer and Telecommunication Systems, Aug. 2001, pp. 125–132.

[10] S. Athuraliya, V. H. Li, S. H. Low, and Q. Yin, “REM: Active queue management,” *IEEE Network*, vol. 15, no. 3, pp. 48–53, May 2001.

[11] W. Feng, D. Kandlur, D. Saha, and K. Shin, “A self-configuring RED gateway,” in *Proceedings of the Conference on Computer Communications (IEEE IN FOCOM)*, vol. 3, Mar. 1999, pp. 1320–1328.

[12] S. Floyd, R. Gummadi, and S. Shenker, “Adaptive red: An algorithm for increasing the robustness of red’s active queue management,” *[Online] Available http://www.icir.org/floyd/papers.html*, Aug. 2001.

[13] “ns (network simulator),” *[Online] Available http://www.isi.edu/nsnam/ns*, 1995.

[14] S. Floyd, “RED: Discussion of setting parameters,” Nov. 1997. [Online]. Available: http://www.icir.org/floyd/REDparameters.txt

```
Initialization:

    $q_i \leftarrow 0$

    $t_i \leftarrow current\_time()$

for each packet arrival:

    $q_i \leftarrow q_{i+1}$

    if ($q_{i+1}$ exceeds $min_{th}$) {

        if ( $q_{i+1} - q_i > slope_{th}(current_t ime() - t_i)$ ) {

            $p_d \leftarrow max_p^r$

        } else {

            /* $p_a$ use the same formula as in RED */

            $p_d \leftarrow p_a$

        }

    }

    $t_i \leftarrow current\_time()$
```

Variables:

q_{i+1}: average queue occupancy (current sample)

q_i: average queue occupancy (previous sample)

t_i: previous sampling time

$current\_time()$: function which return current time

max_p^r: maximum drop probability used in *input rate climbing*

p_d: drop probability used to handle the incoming packet

Figure 1: RARED algorithm

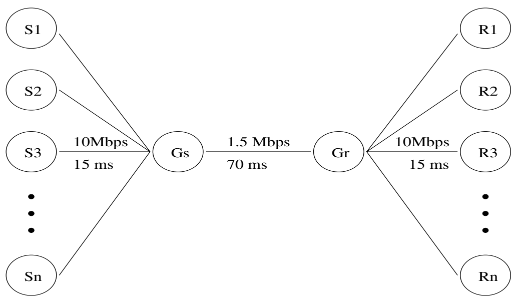

Figure 2: Simulation Network Topology

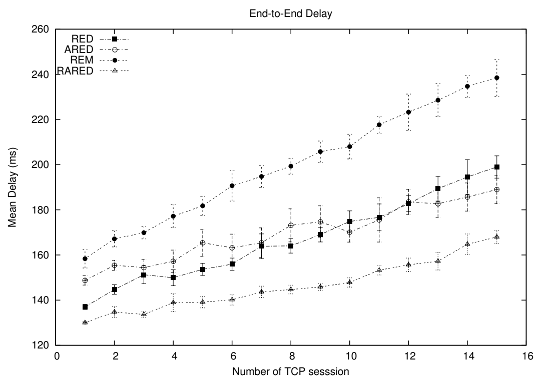

Figure 3: End-to-end mean delay

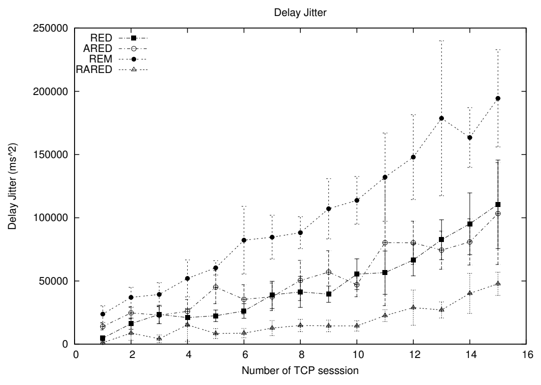

Figure 4: Variance of end-to-end delay

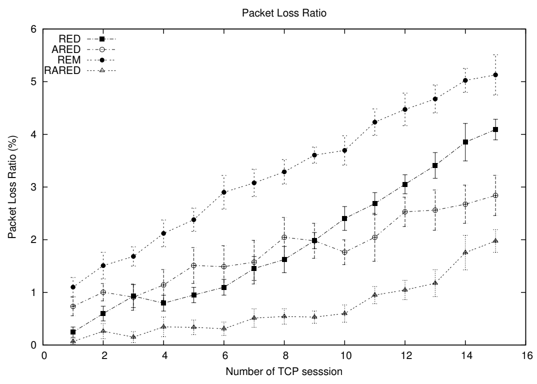

Figure 5: Packet loss ratio

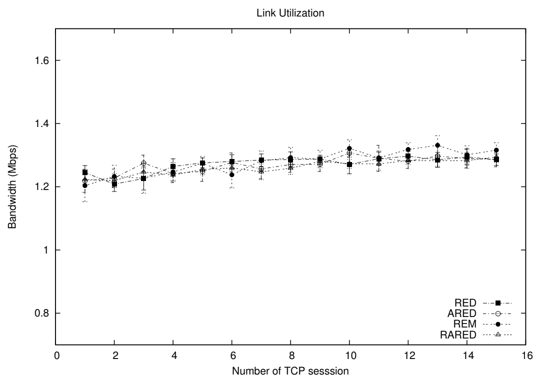

Figure 6: Link utilization