International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:5, No:9, 2011

A Third Drop Level For TCP-RED Congestion Control Strategy

Nabhan Hamadneh, Michael Dixon, Peter Cole, and David Murray

Abstract—This work presents the Risk Threshold RED (RTRED) congestion control strategy for TCP networks. In addition to the maximum and minimum thresholds in existing RED-based strategies, we add a third dropping level. This new dropping level is the risk threshold which works with the actual and average queue sizes to detect the immediate congestion in gateways. Congestion reaction by RTRED is on time. The reaction to congestion is neither too early, to avoid unfair packet losses, nor too late to avoid packet dropping from time-outs. We compared our novel strategy with RED and ARED strategies for TCP congestion handling using a NS-2 simulation script. We found that the RTRED strategy outperformed RED and ARED.

Keywords—AQM, congestion control, RED, TCP.

I. Introduction

Current high speed network gateways are likely to be congested due to the increased demand for the limited network resources such as routers and link bandwidths. Early TCP congestion control strategies attempted to manage congestion by manipulating the congestion window size *cwnd*; which is a parameter that regulates the sending rate [1].

The goals of any congestion control strategy are: (I) Fair resource allocation (II) Reasonable queuing delay (III) minimal packet loss and (IV) Low resource consumption.

These four goals are conflicting. For example, if we design a strategy to reduce the packet loss rate at the gateway, higher queue sizes will be produced. Higher queues on routers increase end-to-end delays. In this paper, we propose the Risk Threshold RED (RTRED) strategy to better balance these conflicting goals.

The key idea of congestion control is to determine the congestion level to start packet dropping. Unfortunately, there is no strategy that provides a perfect balance. In the next sections, we show how RTRED solves some of the imbalances in previous strategies. We also show how RTRED better balances packet loss, average delay and queue space utilization.

This paper is organized as follows: section II introduces TCP congestion control. Section III describes the previous work in TCP congestion control. Section IV introduces the algorithm of our proposed strategy. The network topology used in our simulator is presented in section V. Section VI analyzes the simulation results and section VII concludes our paper.

The authors are with the School of Information Technology, Murdoch University, South St, Murdoch, WA 6150, Australia.

N. Hamadneh, Office Number: ECL-4.042, Phone: +61-4-49096732, Email: n.hamadneh@murdoch.edu.au.

M. Dixon, Office Number: ECL-3.047, Phone: +61-8-93606086, Email: m.dixon@murdoch.edu.au.

P. Cole, Office Number: ECL-3.041, Phone: +61-8-93602918, Email: p.cole@murdoch.edu.au.

D. Murray, Office Number: ECL-3.048, Phone: +61-8-93602723, Email: d.murray@murdoch.edu.au.

II. Background

There are two main approaches for handling congestion in TCP networks. The first approach is congestion recovery and the second is congestion avoidance.

Congestion recovery works after the gateway is overloaded. Strategies that use this approach are source algorithms. These algorithms adjust the sending rate upon congestion signals, such as triple acknowledgments, time-out or Explicit Congestion Notification (ECN). TCP Tahoe [2], TCP Reno [3], [4], and TCP Vegas [5] are examples of these strategies.

In the congestion avoidance approach, some arrangements are made before the gateway is overloaded. Because they are applied by network components, the strategies that apply this approach are called network algorithms. Active Queue Management (AQM) is one of the algorithms that implements this approach [6]. The Random Early Detection (RED) and its variants are the most popular strategies that adopted the AQM algorithm for congestion avoidance [7].

Source algorithms operate end-to-end to send packets at the perfect rate. The congestion window $(cwnd)$ is modified to adjust the sending rate. The manner in which the sending rate is adjusted is called Additive Increase/Multiplicative Decrease (AIMD). When a transmitted segment is successfully acknowledged the window is additively increased. The window is decreased in a multiplicative manner upon packet loss, time out and Explicit Congestion Notification (ECN) signals.

Normally, the congestion recovery and congestion avoidance algorithm work in conjunction to handle congestion. The earlier implementation for network congestion control strategies is the Tail Drop (TD) strategy [6]. This strategy uses First In First Out (FIFO) queue management. When the gateway buffer is overloaded and a packet is dropped, the source algorithm interprets this as a congestion. TCP reacts by reducing the sending rate by adjusting the congestion window size.

Tail Drop implementation has two main problems which are: the Lock Out and the Full Queue problems. The first problem occurs when a few connections monopolize the queue space. The second problem occurs when the gateway keeps sending full queue signals to the sources for a long period of time [1]. The AQM approach implemented by RED and its variants was designed to fix these two drawbacks of TD strategy.

Now, the question is: did RED and its variants really fix these two problems of TD strategy? If yes, what was the price of the solution? This paper tries to answer this question.

III. Previous Work

A. Earlier Congestion Control Strategies

Traditional congestion control policies including; Source Quench, Fair Queuing, No Gateway Policy and Congestion Indication drop packets in the order they arrive. This is similar to the TD strategy and causes similar problems to these described in [8]. One of the TCP traffic characteristics is the burstness. Bursty connections always need more buffer size to absorb their traffic. In addition, they always try to monopolize more than their permitted share of bandwidth and buffer size. When a TCP window of n packets arrives at a TD congested gateway, all the packets of this window will be dropped in the order they arrive. Consequently, n congestion signals will be sent to the same source which is kind of aggressiveness against these types of traffics.

Random Drop (RD) was designed by IETF to avoid the shortcomings of TD. The method of RD is to drop packets randomly rather than from the tail of a queue. In RD, the probability of a TCP flow losing a packet is proportional to the percentage of packets currently occupying the buffer. However, RD has some shortcomings as well. It chooses packets to be dropped by inspecting the buffer distribution only at the time of overflow, disregarding all previous history. Therefore, it unfairly favors the connections with large packets [8].

Early Random Drop (ERD) strategy was designed to fix the problems of RD [8]. At imminent congestion, the gateway begins to drop packets at a rate that is derived from the current network congestion level. If the queue length is greater than the drop level – which is the threshold in RED – then ERD chooses packets randomly. If the probability of this packet is less than a preset drop probability then the packet is to be dropped.

B. RED-Based Congestion Control Strategies

To avoid inspecting the buffer distribution only at the time of overflow, which is a problem of RD, Random Early Detection (RED) inspects the average queue size for the previous history. Also, RED keeps two threshold parameters rather than one in ERD. In addition, the drop probability is dynamically adjusted during the network operation time. For every packet that arrives at the gateway, RED calculates the average queue size using (1). If the average is between the minimum and the maximum thresholds then the arriving packets will be dropped with probability p_a which is calculated in (3). If the average is greater than the maximum threshold then every arriving packet will be dropped with probability p_a. Otherwise, the average is less than the minimum threshold and no packet has to be dropped. Equation (2) calculates the immediate dropping probability p_b which is a parameter used to calculate p_a.

$$avg = (1 - w_q) * avg + w_q * q \tag{1}$$

$$p_b = max_p(\frac{avg - min_{th}}{max_{th} - min_{th}}) \tag{2}$$

$$p_a = p_b(\frac{1}{1 - count * p_b}) \tag{3}$$

Where:
avg : *Average queue size.*
w_q : *A weight parameter,* $0 \leq w_q \leq 1$.
q : *The current queue size.*
p_b : *Immediately marking probability.*
max_p : *Maximum value of* p_b.
min_{th} : *Minimum threshold.*
max_{th} : *Maximum threshold.*
p_a : *Accumulative drop probability.*
$count$: *Number of arrived packets since the last dropped one.*

The network flows which react to congestion, such as TCP flows, are responsive flows. Flows that do not adapt the sending rate based on congestion conditions are unresponsive; such as UDP flows [9]. Unresponsive flows can occupy more than their allowed share of network resources.

Flow Random Early Detection (FRED) preferentially discards packets from responsive and unresponsive flows. RED-DT is a per-flow strategy that distributes the buffer space fairly between responsive and unresponsive flows [10]. Logest Queue Drop (LQD) [11] is a strategy that follows this approach.

Dynamic And Self-Adaptive TCP-Friendly Congestion Control Mechanism (DATFCC) adjusts the dropping probability relating to the type of flow and the buffer usage ratio [12]. It uses the TCP friendly approach [13], [14] to maintain fairness between TCP flows and real-time UDP flows.

RED Optimized Dynamic Threshold (RED-ODT) uses the DT Scheme for shared buffer management [15]. It adjusts the maximum threshold and minimum threshold relating to the actual queue size and the buffer size in multi-queue gateways.

Adaptive-RED (ARED) increases and decreases the max_p parameter. When the average queue size is greater than the maximum threshold, max_p is increased. When the average queue size is less than the minimum threshold, max_p is decreased [16].

IV. RTRED Strategy

In this section, we describe the approach and scenarios that motivated the design of RTRED strategy.

A. RTRED Motivations

New congestion control policies tend to assign the congestion level for traffic management and congestion recovery processes. Rather than using the actual queue size as the congestion level indication, like TD strategy does, RED and RED-based strategies keep an Exponentially Weighted Moving Average $(EWMA)$ queue size to detect the congestion level. If the average queue size avg exceeds the max_{th} parameter then the gateway has reached an extreme congestion level and all incoming packets are to be dropped with probability p_a. If avg is in-between the min_{th} and max_{th} then the gateway is experiencing congestion. Arriving packets are to be dropped or marked with probability p_a. If avg is less than the min_{th} then there is no congestion and no packet has to be dropped.

RED was initially designed to minimize packet loss and queuing delays. It was also designed to maintain high link utilization and to remove biases against bursty traffic. Furthermore, it tries to avoid global synchronization which occurs when all network resources reduce their sending rate at the same time.

The problem with RED is the mismatch between the macroscopic and microscopic behaviors of queue length dynamics. This malfunction occurs when a peak in the actual queue size (microscopic behavior) exceeds the available buffer size. The average queue size is still low (macroscopic behavior) whereas the actual queue size is very high. This is caused by a small weight parameter w_q in (1). As a result, a congestion signal is detected by network sources due to time outs caused by packet drops in the router. This means that the TCP source algorithm, such as Reno, is responsible for congestion handling, whereas the network algorithm, which is a RED-based strategy, behaves like TD. The authors in [1], describe this problem as the mismatch between microscopic and macroscopic behavior of queue length dynamics.

Fig. 1a and Fig. 1b illustrate the two possible scenarios of this mismatch. In phase 1, packets will be dropped due to time out signal. In phase 2 packets will be dropped unfairly because of the high value of avg parameter whereas the actual queue size is less than the min_{th}.

Choosing the drop level and the drop probability is the greatest challenge of RED and RED-based strategies. ARED proposed to multiply the max_p parameter by another parameter; $alpha$, when avg exceeds the max_{th}, and to divide it by $beta$ when avg is less than the min_{th} parameter. An additional strategy named Gentle RED proposed to vary the drop probability from max_p to 1 when the average queue size varies from max_{th} to twice the max_{th} parameter [17], [18], [19].

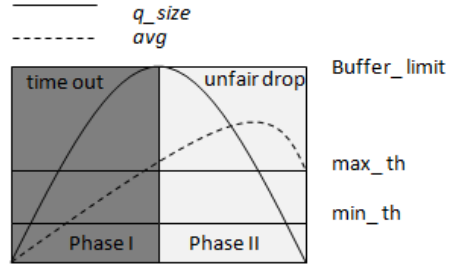
Many RED variants have been proposed and the mismatch between the microscopic and macroscopic behaviors persist. In the next section, we propose the Risk Threshold RED (RTRED) strategy which provides more timely congestion indication. RTRED reduces the amount of unfairly dropped or timed out packets.

B. RTRED Algorithm

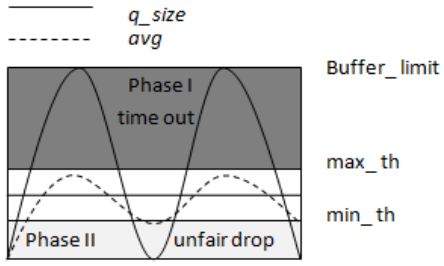
The drop level defines the safe area of traffic fluctuation. Hence, drop level should alert the aggressive connections to slow down before the gateway is overloaded. It should be small enough for sources to respond before gateway overflow. Conversely, small threshold values would cause false congestion panic and unnecessary losses. Therefore, the threshold must be dynamically readjusted depending on the current network traffic.

The drop probability also, should be chosen carefully. It should be large enough to detect the misbehaved connections and small enough to protect the well behaved ones. Therefore, the drop probability should be adjusted dynamically as well.

TD strategy defines the drop level depending on the actual queue size. In addition to the global synchronization phenomenon, this scenario would lead to another two problems which are: the Lock Out and Full Queue phenomena. RED strategy uses only the weighted average parameter to define the drop level. This sometimes leads to unfairly dropped or timed-out packets.



(a) scenario I.



(b) scenario II.

Fig. 1 Time out and unfair drop scenarios

In order to remedy the shortcomings of RED and TD strategies, RTRED uses both the actual and the average queue sizes to define the drop level.

Fig. 2 illustrates our proposed RTRED algorithm. In this algorithm, we add an extra drop level, which is the risk threshold. If the actual queue size exceeds this risk threshold then we use the initial max_p parameter to calculate the drop probability. This will reduce the number of packets lost due to timeout signals. If the actual queue size is less than the min_{th}, then max_p parameter is reduced five times the initial max_p. This in turn will minimize the amount of unfairly dropped packets at the gateway.

When the average and the actual queue sizes are in between the maximum and the minimum thresholds, RTRED halves the max_p parameter. This allows the queue to safely increase without any risk of congestion. By doing so, RTRED reduces packet losses and better utilizes the queue space. It also removes aggressiveness against short lived bursty traffic.

V. Network Topology

In our simulator we use a network topology with six nodes sharing the same bottleneck link. The start time for nodes is uniformly distributed between 0 and 7 seconds with a 552 bytes packet size. The link between each node and the gateway is a duplex link with $10Mb$ capacity and a uniformly distributed delay between $1ms$ and $5ms$. The bottleneck link between the gateway and the sink is a duplex link with $0.7Mb$ capacity and $20ms$ delay. Fig. 3 illustrates this network topology.

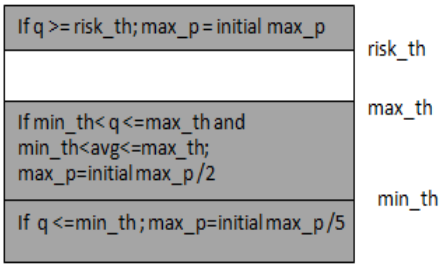


Fig. 2 RTRED algorithm

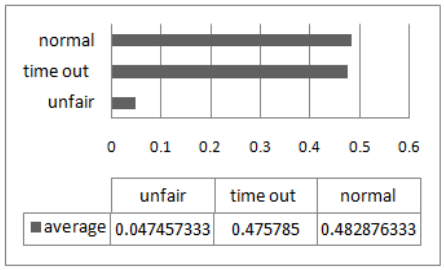


Fig. 5 Total percentage of packets dropped for each drop area

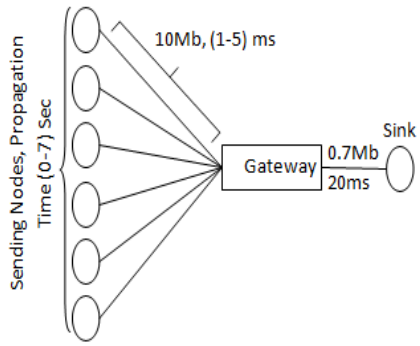


Fig. 3 Network topology.

TABLE I Parameter configuration.

Scenario	min_{th}	max_{th}	max_p	$risk_{th}$	$Buf.$
1	12	25	0.05	28	30
2	60	90	0.1	95	100
3	15	30	0.08	40	50

VI. Simulation and Analysis

In this work we use a NS-2 script to simulate three different scenarios for RED, ARED and RTRED. Table I, illustrates the parameter configuration for these scenarios.

In Fig. 2 we divide the gateway queue into four areas. The white area in the table depicts the safe area in which the queue can fluctuate safely without any congestion indicator to absorb bursty traffic. The area between the $risk_{th}$ and the buffer limit is the time out area in which the risk of a packet drop due to time out signal is very high. The normal drop area is the area between the min_{th} and max_{th}. RED and its variants normally start dropping packets in this area. Finally, any packet dropped in the area between the empty queue and the min_{th} is the unfair drop area.

In Fig. 4 we trace the percentage of packets that have been dropped in every area for each scenario separately. We see that the majority of the packets have been dropped in the normal area and the time out area. Also, a few packets have been dropped in the unfair area. That explains why we defer using the high initial max_p parameter until the actual queue size exceeds the $risk_{th}$. The bar chart in Fig. 5 depicts the average percentage of packets dropped for the three scenarios.

To avoid the mismatch between the microscopic and macroscopic behaviors of queue dynamics, RTRED works as a compromise between TD and RED strategies. It only checks the actual queue size in the Timeout and Unfair areas to avoid buffer overflow and unfair packet losses respectively. In order to reduce the drop rate and avoid wasting the queue space, it checks the average and the actual queue sizes in the normal area, controlling congestion by reducing max_p to half of the initial parameter.

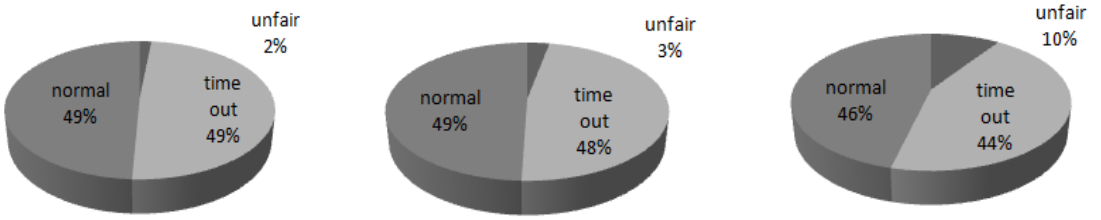
TABLE II Network performance for the three scenarios.

Scenario	Strategy	Ave Q Size (Packet)	Average Delay Time (ms)	Loss Rate (Packet)	No. of Propagated Packets
1	RED	20.292	122.85184	710	18158
	Adaptive	16.469	123.83286	855	18303
	RTRED	21.2004	122.594742	668	18120
2	RED	54.4667	119.340434	151	17639
	Adaptive	60.9756	119.387794	147	17646
	RTRED	62.1799	119.259245	136	17627
3	RED	21.0212	122.527085	651	18110
	Adaptive	19.8217	122.80448	698	18151
	RTRED	24.8899	121.816685	548	18005

Table II, illustrates the network performance for the three scenarios. It is clear from the table that RTRED strategy has outperformed RED and ARED strategies. It maintains a higher average queue size; increasing the gateway queue utilization. In response, this will reduce the packet loss rate which is not conflicted with the small average delay time values for RTRED. This technique reduces the total amount of packets propagated by the source nodes which in turn reduces the overhead of packet retransmission. The average delay time is calculated for the actual amount of packets queued at the gateway rather than the average queue size value.

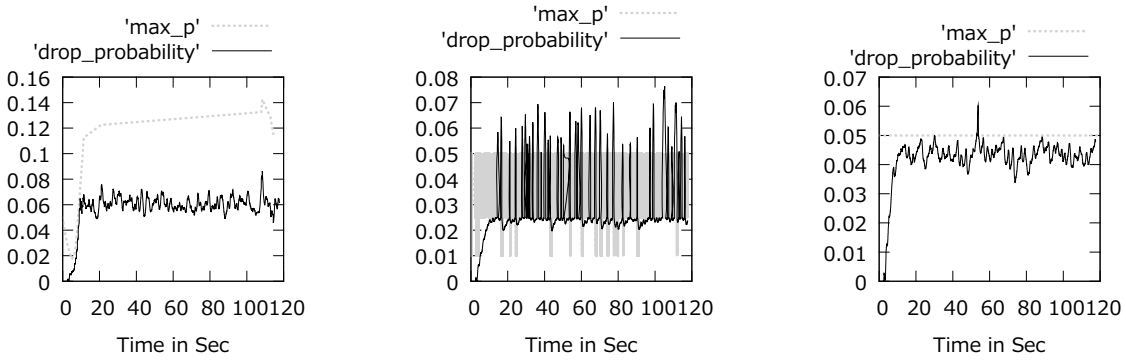
Fig. 6 and Fig. 7 illustrate scenario one simulation results. Fig. 6, depicts the drop probability and max_p parameters for the three strategies. In this figure we see how max_p parameter for RTRED fluctuates between the initial max_p parameter and $max_p/5$ value, depending on the queue size dynamics. In response, the drop probability increases and decreases dynamically to fit the current congestion level. On the other hand, ARED keeps high max_p parameter which is unresponsive to the current congestion level. This is illustrated in Fig. 6a. RED also keeps a fixed max_p parameter, which is unresponsive to the queue dynamics as Fig. 6c shows.

Fig. 7 plots the average and the actual queue sizes for the three strategies. In Fig. 7b, we see how RTRED strategy allows the queue to increase with less dropping rate. It stabilizes



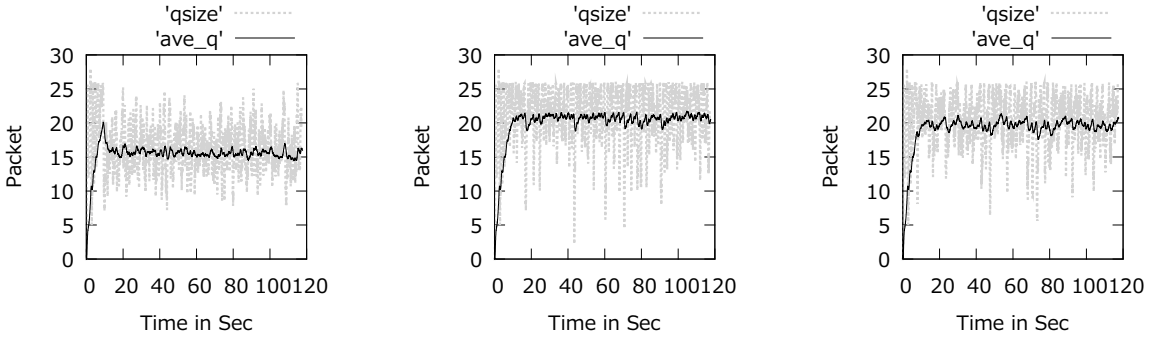
(a) Scenario I. (b) Scenario II. (c) Scenario III.

Fig. 4: Percentage values for packets dropped in each drop area.



(a) Adaptive-RED. (b) RTRED. (c) RED.

Fig. 6: Drop probability and max_p parameters for scenario I.



(a) Adaptive-RED. (b) RTRED. (c) RED.

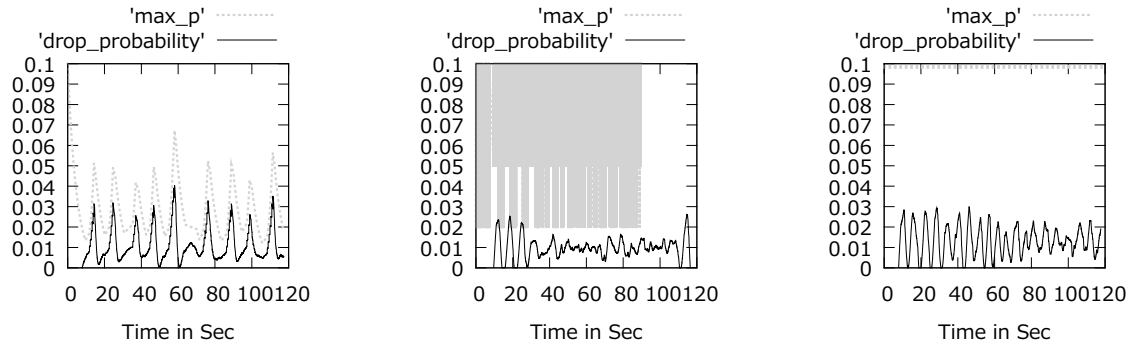
Fig. 7: Average and actual queue sizes for scenario I.

the queue size at the safe area with no risk of unnecessarily drop due to time out signal. Fig. 7a and Fig. 7c are the figures of ARED and RED queue dynamics respectively. The figures show how aggressively and unfairly these strategies drop packets to avoid buffer overflow and to minimize the average queue size.

In this scenario, RTRED provides accurate calculation of the drop level and the drop probability. In fact, it reflects high trustworthy congestion indication before network starts recovering.

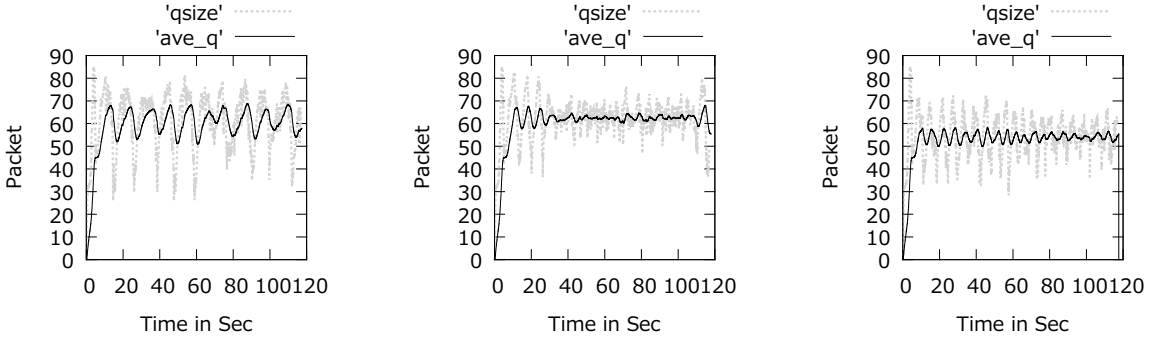
RED gateways are full of tradeoffs. The tradeoff between decreasing delay and increasing throughput is one of the major problems that appears at the time of configuring the drop thresholds. Another tradeoff between link utilization and average delay time will make the problem worst. Larger min_{th} values will increase link utilization which is a good performance behavior, but in [7] it is suggested to set max_{th} twice the min_{th} which will increase delay.

The idea of small queues is proposed by [7] to avoid long delays. RTRED shows that this is not always right, because, the desirable queue size is the size that helps the gateway reduces the drop rate and the overhead of packet retransmission. The average and actual queue sizes of RTRED are higher than RED and ARED. The actual end-to-end delays of RTRED,



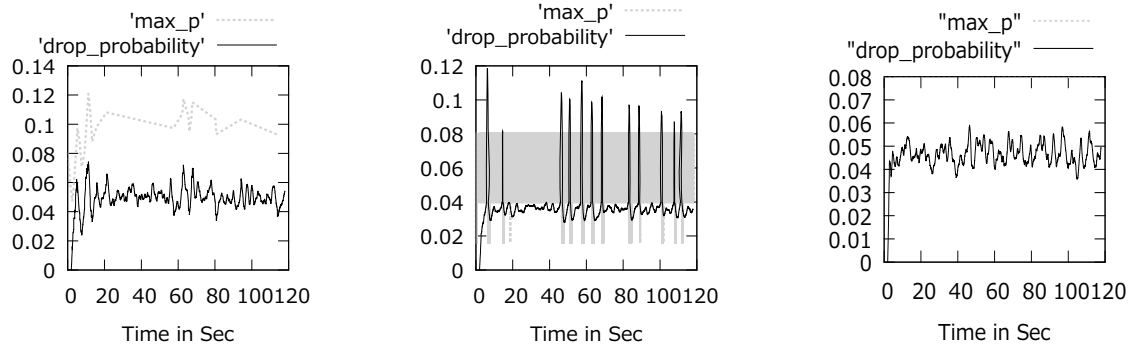
(a) Adaptive-RED. (b) RTRED. (c) RED.

Fig. 8: Drop probability and max_p parameters for scenario II.



(a) Adaptive-RED. (b) RTRED. (c) RED.

Fig. 9: Average and actual queue sizes for scenario II.

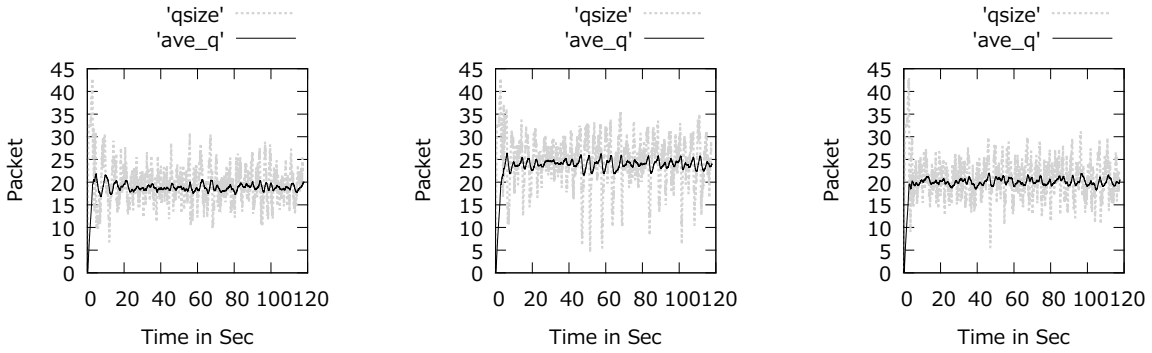


(a) Adaptive-RED (b) RTRED. (c) RED.

Fig. 10: Drop probability and max_p parameters for scenario III.

shown in table 2, are lower due to fewer retransmissions.

Fig. 8 – Fig. 11 illustrate how RTRED outperforms RED and ARED for scenarios II and III respectively. The figures show the drop probability and the queue dynamics for each strategy. RTRED provides the most stable queue length with lower packet drops. It also maintains very dynamic max_p parameter and drop probability which are more responsive to the queue dynamics. RTRED would not increase the drop rate unless a strong signal of congestion is detected. Regardless of having a dynamic max_p parameter in ARED, the strategy has a problem in adjusting this parameter to accommodate the queue size fluctuations.



(a) Adaptive-RED. (b) RTRED. (c) RED.

Fig. 11 Average and actual queue sizes for scenario III

VII. Conclusion

This work proposes a novel RED-based strategy; Risk Threshold RED (RTRED), which is designed to avoid the mismatch between the microscopic and macroscopic behaviors of queue length dynamics. The proposal is a compromise between the TD and RED strategies for congestion handling in TCP networks.

TD uses the actual queue size to define the congestion level. RED uses the Exponentially Weighted Moving Average to define the congestion level. RTRED uses both the actual and the average queue sizes to calculate the drop probability and the congestion level. These calculations operate in conjunction with a third drop level; the risk threshold.

Using an NS-2 simulation, The results suggest that RTRED outperformed competing strategies; reducing the unnecessary packet loss rate, the average delay time and the overhead of packet retransmission. Furthermore, RTRED avoids wasting the gateway buffer size and increases the buffer utilization.

References

[1] S. Ryu, C. Rump and C. Qaio, "Advances in Internet Congestion Control, IEEE Communications Surveys," *The Electronic Magazine of Original Peer Reviewed Survey Articals*, vol. 5, 2003.

[2] V. Jacobson, "Congestion avoidance and control," *SIGCOMM '88: Symposium proceedings on Communications architectures and protocols*, 1988, pp.314–329, Stanford, California, United States, http://doi.acm.org/10.1145/52324.52356, ACM, New York, NY, USA.

[3] M. Allman, V. Paxson and W. Stevens, "TCP Congestion Control,", *IETF* RFC 2414, September 1998.

[4] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," *IETF* RFC2001, 1997.

[5] L. Brakmo, and L. Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet," *IEEE Journal on Selected Areas in Communication*, vol. 13, no. 8, pp. 1465-1480, October 1995.

[6] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, *et al*, "Recommendations on Queue Management and Congestion Avoidance in internet," *IETF* RFC2309, 1998.

[7] S. Floyed and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transaction on Networking*, vol. 1, no. 4, pp. 397 – 413, 1993.

[8] E. Hashem, "Analysis of Random Drop for Gateway Congestion Control," *M.I.T Laboratory for Computer Science 465*, 1989.

[9] L. Vukadinovic and L. Trajkovic, "RED With Dynamic Thresholds For Improved Fairness," *ACM Symposium on Applied Computing*, 2004.

[10] D. Lin, and R. Morris, *Dynamics of Random Early Detection*, in ACM SIGCOMM '97, pp. 127-137, Cannes, France, 1997.

[11] B. Suter, T. V. Lakshman, D. Stiliadis, and A. K. Choundhury, "Design Considerations for Supporting TCP with Per-flow Queuing," *INFOCOM '98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies, Proceedings*, pp. 299 - 306 vol.1, San Francisco, CA, 1998.

[12] L. Wei-yan, Z. Xue-lan, L. Tao, and L. Bin, "A Dynamic and Self-Adaptive TCP Friendly Congestion Control Mechanism in Next-Generation Networks," *Intelligent Systems and Applications ISA*, International Workshop, pp. 1-4, Wuhan, 2009.

[13] J. Padhye, V. Firoiu, D.F. Towsley, and J. F. Kurose, "Modeling TCP Reno Performance: a Simple Model and its Empirical Validation," *Networking, IEEE/ACM Transactions*, pp. 133 - 145, vol. 8 no. 2, 2000.

[14] D. Bansal, and H. Balakrishnan, "Bionomial Congestion Control Algorithms," *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, Proceedings*, pp. 631 - 640, vol. 2, Anchorage, AK, 2001.

[15] H. Chengchen, and L. Bin, "RED With Optimized Dynamic Threshold Deployment on Shered Buffer," *Advanced Information Networking and Applications AINA, 18th International Conference*, pp. 451 - 454, 2004.

[16] W.-C. Feng, D.D. Kandlur, D. Saha, and K. G. Shin, "A Self-Configuring RED Gateway," *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Proceedings*, pp. 1320 - 1328, vol. 3, New York, NY, 1999.

[17] S. Floyd, *Recommendation on using the gentle-variant of RED*, http://icir.org/floyd/red/gentle.html, 2000.

[18] V. Rosolen, O. Bonaventure, and G. Leduc, "A RED discard strategy for ATM networks and its performance evaluation with TCP/IP traffic," *SIGCOMM Comput. Commun. Rev. ACM*, pp. 23-43, vol. 29, no. 3, New York, NY, USA, 1999.

[19] V. Rosolen, O. Bonaventure, and G. Leduc, "Impact of cell discard strategies on TCP/IP in ATM UBR networks," *6th Workshop on Performance Modelling and Evaluation of ATM Networks (ATM'98)*, Ilkley, UK, 1998.