QSRED, An Algorithm To Solve The Mismatch Between The Microscopic and Macroscopic Behavior of RED Gateways

Nabhan Hamadneh[†], David Murray[††], Michael Dixon[†††] and Peter Cole[††††]

[†]School of IT, Murdoch University, ECL 2.057, South Street, Murdoch , WA 6150, Mob: +61 449096732
[††]School of IT, Murdoch University, ECL 3.048, South Street, Murdoch, WA 6150, Phone: +61 8 9360 2723
[†††]School of IT, Murdoch university, ECL 3.047, South Street, Murdoch, WA 6150, Phone: +61 8 9360 6086
[††††]School of IT, Murdoch University, ECL 3.041 South Street, Murdoch, WA 6150, Phone: +61 8 9360 2918

Summary
Network congestion is a phenomenon caused by the extreme demand of restricted network resources. Various congestion control strategies have been proposed to increase network performance. This study suggests that there is a mismatch between the microscopic and macroscopic behavior in (Random Early Detection) RED's queue management mechanism. This work investigates this problem and propose QSRED (Queue Sectors RED) to avoid unsatisfactory performance. QSRED is simulated against RED and ERED (Effective RED) by measuring: throughput, link utilization, packets loss and average delay using the NS2 simulator. The results suggest that Queue Sectors RED (QSRED) helps RED overcome the mismatch between microscopic and macroscopic behavior of queue length dynamics.

Key words:
TCP , Congestion, RED, AQM, QSRED.

1. Introduction

Congestion control is subject to design strategies and algorithms that can dynamically control traffic sources when demand exceeds the available capacity. Random packet dropping is one of the earliest techniques used for congestion handling [1]. Current Internet congestion handling strategies are used to improve performance.
Many congestion control approaches have been proposed in the literature [2]. Active Queue Management (AQM) is an algorithm executed by network components, such as routers, to detect and inform senders of congestion. In AQM, routers will actively drop packets from queues to signal that the sender should slow down [3]. Random Early Detection (RED) is an AQM strategy which was initially designed to minimize packets loss and queuing delays. AQM strategies should also maintain high link utilization and remove biases against bursty traffic [4, 5].
This work discusses the mismatch between the microscopic and macroscopic behavior of queue length dynamics in AQM strategies, particularly RED-based strategies. The macroscopic behavior of a queue is the average queue size. The average queue size is not the mean of the actual queue size. Rather, it is a function of a weighted parameter w_q which illustrated in Eq.(1). Smaller w_q parameters exacerbate the mismatch problem.
The microscopic behavior of a queue reflects the actual queue size. AQM strategies work in conjunction with the TCP protocol to control congestion. In some scenarios, actual queue sizes exceed the available buffer size, causing packets to be dropped. Packet loss is a congestion indicator used by TCP, not by the AQM strategy.
At the same time, the AQM strategy in some situations could not realize the problem because it is working with the average queue size. It does not recognize the current peak in the actual queue size.
In this work we add an algorithm to the traditional RED implementation to minimize the problems associated with this mismatch. Our proposal is tested with the four network performance parameters which are: throughput, link utilization, packets loss and average delay.
This paper is organized as follows: firstly, we survey TCP congestion control in section 2. In sections 3 and 4, we describe RED's and ERED's implementations respectively. Section 5 describes the mismatch problem between queue length dynamics. Then we propose the QSRED algorithm in section 6. Section 7 present the network topology of our simulator. In section 8, we present the network performance parameters for our algorithm. Section 9 concludes our paper.

2. Background

2.1 Transport Control Protocol (TCP)

TCP is the transport control protocol which is responsible for end-to-end data transmission between nodes in current networks. It was originally established by the Internet Engineering Task Force (IETF) organization. The TCP protocol is the most widely used protocol in real networks. About 90% of the current networks rely on this protocol for data delivery [6].

Some of the TCP objectives are:

Manuscript received November 5, 2010
Manuscript revised November 20, 2010

(i) Network congestion control.
(ii) Retransmission of lost packets.
(iii) Adapt data transmission to the available network bandwidth.

In TCP, each transmitted packet has a sequence number. For every successfully delivered data packet, the receiver sends an ACK signal with the packet sequence number to acknowledge that the packet has been received intact. Rather than sending packets one by one, TCP protocol sends packets in groups. The maximum number of packets allowed in each group is called the congestion window size cwnd. This number should not exceed the available bandwidth of the network. In the steady state, the rate of sending packets will match the rate of ACKs received by the sender. Acknowledgments regulate the sending rate. They also provide reliability by informing the sender of lost packets enabling retransmission.

2.2 TCP Congestion Control

The ACK informs the source about the sequence number of the next packet which is expected to be received. There are two methods to warn the sender of lost packet.

The first method is the Implicit Congestion Notification ICN. Suppose a sender with n+2 packets to be sent. When the destination node receives the n^{th} packet it sends an Implicit ACK that it is expecting the packet number n+1. If this packet has been lost in the network and the packet number n+2 has arrived, then the receiver sends a duplicate ACK to inform the sender that packet n+1 is still missing. If three duplicate ACKs arrive at the source which is called the triple acknowledgment, then the packet is considered lost. Another approach to indicate lost packet is the timeout signal. In this approach, the source has a timer for every sent packet, if the ACK does not come back from the receiver within time T then the packet considered lost. These are both forms of Implicit Congestion Notification.

The second method for congestion notification is the Explicit Congestion Notification (ECN). When a gateway becomes congested, it can send a packet to the TCP sender with the ECN bit set. This informs the sender that it should slow down [7].

The TCP source node adjusts the congestion window based on these congestion signals. It decreases the congestion window when the level of congestion goes up and increases the congestion window when the level of congestion goes down. Altogether, the mechanism is commonly called additive-increase/multiplicative-decrease [2, 8].

Slow start is a mechanism to prevent immediate congestion state. It sets the cwnd parameter to 2 and starts increasing the congestion window size exponentially every time an ACK arrives to the source. When source stops receiving ACKs this indicates congestion and the cwnd parameter goes back to 2 [9].

Packet retransmission after a triple acknowledgment is called fast retransmit because the node does not wait for a time out signal to retransmit the lost packet. In case of congestion, rather than reducing the cwnd to 2 in slow start phase, it is better idea to halve the cwnd to increase the network throughput. This mechanism is called fast recovery. The preceding mechanisms are called fast retransmit and fast recovery or Reno [10].

2.3 Active Queue Management (AQM) Approach

All TCP variants are congestion recovery mechanisms. More specifically, They come into play after congestion occurs and the buffer is already overflowed. There is another approach for handling congestion which is the Active Queue Management AQM [11].

Accordingly, the design of a congestion control strategy consists of two parts: a network algorithm such as RED and a source algorithm such as Reno. The network algorithm is responsible for preemptively detecting congestion and informing the source of the congestion. In response, the source algorithm adjusts the sending rate by reducing the congestion window size [6, 12]. The network algorithms, such as RED, operate on the intermediate routers between the source and the destination. The source algorithms, such as TCP, operate at the source and destinations of the traffic flow. This study proposes an improvement to RED.

3. Random Early Detection (RED) Strategy

3.1 RED Design Objectives

RED maintains an Exponentially Weighted Moving Average (EWMA) of the buffer size on internet routers [5]. Equations 1, 2 and 3 illustrate how the drop rates of packets are calculated.

$$avg = (1 - w_q) * avg + w_q * q \quad (1)$$

$$P_b = max_p \left(\frac{avg - min_{th}}{max_{th} - min_{th}} \right) \quad (2)$$

$$p_a = p_b \left(\frac{1}{1 - count * p_b} \right) \quad (3)$$

where:

avg : average queue size

w_q : a weight parameter, $0 \le w_q \le 1$

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 probability

count: packets since last marked packet

The RED gateway has two preset threshold values which are the maximum and the minimum thresholds. Every time a new packet arrives at the gateway, the avg value is calculated. If this value is greater than the maximum threshold, then all incoming packets must be marked or dropped. If it is less than the minimum threshold, the arriving packet enters the queue without marking or dropping. When this avg value is in between the minimum and the maximum thresholds, then incoming packets will be dropped or marked with probability p_a [4].

In this way, RED achieves the following:

(i) Connections with higher input rates receive proportionally more drops (or marks) of packets than connections with lower input rate.
(ii) Maintains an equal rate allocation.
(iii) Removes biases against bursty traffic.
(iv) Eliminates global synchronization.

For more details in RED design principles see [13, 14]

3.2 Issues with RED Parameter Configuration

One of the simplest implementations of congestion detection is the Tail Drop (TD) strategy. It defines a drop level. Whenever a queue size exceeds this drop level, the gateway starts dropping packets in the order they arrive which is a First-In First-Out queue management. This implementation has numerous disadvantages. A complete study of these problems is outside the scope of this work but we mention two of the major problems associated with this implementation. The first is the lock out problem. This occurs when a few nodes monopolize the whole network bandwidth [6]. The second problem is the full queue problem. This occurs when a gateway continually sends full queue signals to sources for an extended period of time [6].

AQM strategies, such as RED, were proposed to overcome TD drawbacks. RED maintains two drop levels which are the maximum and minimum thresholds. In order to control congestion efficiently, these two drop levels work in conjunction with the RED parameters described in section 3.1.

Parameter configuration in RED is very difficult[15]. For example, if the difference between the maximum and minimum threshold is too small then the strategy is approaching the TD implementation. In the same manner if this difference is too big, RED cannot recognize the queue dynamics, which can cause the queue to overflow. Parameter setting also depends on the number of active connections, buffer space limitations and the severity of congestion.

It is the same for w_q parameter. Small values for this parameter lead to a mismatch between microscopic and macroscopic behavior of queue length dynamics.

The drop probability p_a is a function of the max_p parameter. The max_p parameter is a constant in the traditional RED implementation. The bigger max_p value the higher p_a parameter. Resulting in more packet drops. Many RED variants have proposed to use a dynamic max_p parameter [6, 16].

3.3 RED Variants

Many RED-based based strategies have been proposed since the original RED proposal. These strategies are: ARED, Blue-RED and ERED.

ARED dynamically adjusts the max_p parameter in Eq.(2) [16]. It increases max_p when avg exceeds max_{th} and decreases max_p when avg goes below min_{th}.

BLUE-RED increases the packet drop probability in response to buffer overflow and decreases the packet drop probability when the link becomes idle [17].

ERED strategy is a modification of RED. In this strategy, max_{th} and min_{th} parameters are subject to change during network operation. In this work we compare QSRED with ERED and traditional RED [18].

4. Effective RED (ERED) Strategy

The recently proposed ERED strategy adjusts max_{th} and min_{th} parameters. Eq.(4) and Eq.(5) illustrate how ERED sets those parameters.

$$max'_{th} = 2 * max_{th} \quad (4)$$

$$min'_{th} = \frac{max_{th} + min_{th}}{2} + min_{th} \quad (5)$$

If $(min'_{th} < avg < max'_{th})$ & $(qlen > min'_{th})$, ERED drops arriving packets with probability p_a. That is to match the current queue size with the average queue size.

If $(avg < min'_{th})$ & $(qlen > 1.75\, max_{th})$ the strategy drops arriving packets with probability p_b. If $(avg > K)$ & $(qlen < T)$ then $avg = 2*(max_{th} + min_{th})/3 + min_{th}$. K and T evaluated by Eq.(6) and Eq.(7).

$$K = \frac{max_{th} - min_{th}}{2} + min_{th} \quad (6)$$

$$T = \frac{max_{th} + min_{th}}{2} + max_{th} \quad (7)$$

5. The Mismatch Between Microscopic and Macroscopic behavior of RED

In RED gateways, we refer to the average queue size as the macroscopic behavior of that queue because it reflects the long term dynamics of the queue. Conversely, the microscopic behavior of queue reflects the short term dynamics of queue. Prior research has shown large differences between the average and the actual queue dynamics [19-21].

Problems occur when a burst of traffic arrives at already congested gateway. If RED maintains a small w_q parameter then the average queue size will be slightly increased. As a result, the gateway buffer overflows and packets are dropped. Congestion will be detected by TCP. After congestion returns to normal levels, average queue sizes will increase. This can lead to unnecessary packet drops. These unnecessary drops cause the congestion window to be reduced far below the optimal level. Fig. 1 depicts this problem.

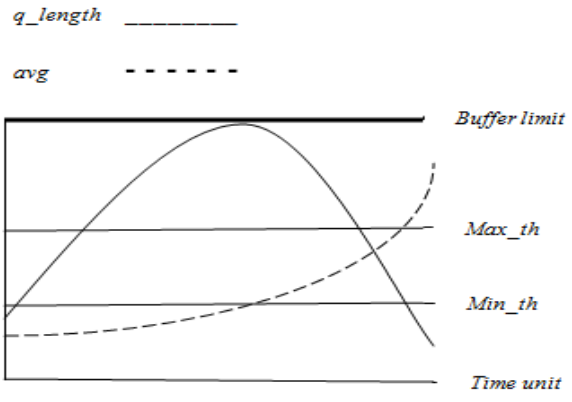

Fig. 1 The mismatch between macroscopic and microscopic of queue length dynamics

6. QSRED Algorithm

Our approach in QSRED is to divide the gateway buffer into six equal sectors. The drop probability and max_p parameters are to be adjusted when the actual and average queue sizes traverse between these sectors. In this section, we show how this approach improves network performance and stability.

ERED added adjustable max_{th} and min_{th} parameters; which matches the microscopic and macroscopic behaviors of the queue in two serious situations. However, ERED still has some drawbacks:

(i) ERED is using the normal parameters of original RED. It is clear from Eq.(4) and Eq.(5) that max'_{th} and min'_{th} are functions of the normal max_{th} and min_{th} and those parameters have the same configuration problem as RED [15].

(ii) The strategy would be more powerful if the max_{th} and min_{th} are adjustable during the simulation run time in response to the traffic load dynamics.

(iii) ERED proposed to keep low loss rate values but does not make any calculation for the other three performance factors: throughput, link utilization and delay.

(iv) If inequality 1 is satisfied, ERED uses the immediate marking probability, which is a function of *avg* and max_p. The value of max_p in this case, is minimally smaller, resulting in a lower dropping rate. Hence, the risk of buffer overflows is high. In such a serious situation, congestion control strategy should increase the dropping probability to a value approaching 1. This will allow the buffer to be drained before it becomes overloaded.

(v) Equally, if the queue length is greater than the drop level $(1.75\, max_{th})$ then it drops the arriving packets with probability P_b. It is also clear from the previous literature that there are no suggestions about the optimal setting of this drop level. Consequently, we propose the following parameter configuration to enhance ERED's functionality:

Table 1: QSRED algorithm

*** for every arriving packet*

if (q_size >= Sec.5 and avg <= Sec.2) Then

$max_p = 2 * max_p$

Else

Go To: Traditional RED Implementation

Firstly, one of the best ways to avoid using max'_{th} and min'_{th} as functions of the actual max_{th} and min_{th} is to choose some parameters related to the buffer size. There is no adequate way to set the max_{th} and the min_{th} of a queue. However, if we are intending to use ERED, it is advantageous to use smaller min_{th} and max_{th} parameters to avoid exceeding the buffer size when calculating max'_{th} and min'_{th} parameters.

Secondly, we have an option to divide the buffer into 6 equal sectors. Then, a reasonable value for the max'_{th} could be *4/6* buffer size, keeping the remaining *2/6* buffer size for short lived bursty traffic. max_{th} can be equal to *3/6* (half the buffer size). min'_{th} and min_{th} can be *2/6* and *1/6* buffer size respectively.

Finally, to avoid the drawbacks (iv) and (v) of ERED, it is advisable to increase the drop probability to help the gateway drop the overloading packets. If the inequalities 1 and 2 are satisfied then we set max_p to $2^* max_p$. In addition, we replace inequality 1 by inequality 3 to execute the actual queue size check. Consequently, we are replacing the parameter $1.75\ max_{th}$ in inequality 2 of ERED by the parameter $5/6 * Buffer - Size$. Also, we use the normal dropping probability p_a rather than using the current small dropping probability p_b. This has the effect of speeding up the queue drain process in case of congestion. Table 1 shows the algorithm of QSRED.

$$(qlen > 1.75\, max_{th}) \quad (1)$$

$$(avg < min'_{th}) \quad (2)$$

$$(ERED - Queue > 5/6 * Buffer - Size) \quad (3)$$

Consider, as an example, a buffer with size 90 packets. Suppose that, *avg=25*, *qlen=80* and the initial $max_p = 0.01$. Then, both of the inequalities 1 and 2 are satisfied. ERED's parameter configuration will be as in Table 2. In this situation, the queue is rapidly accumulating and the buffer is about to be overloaded while ERED keeps a small max_p value. This max_p value is not big enough to regulate the queue size. For better queue management performance, we have to increase the drop probability parameter by a value that allows the queue to return to the normal level (see P_b, for the QSRED) .

QSRED monitors the actual and the average queue size values. If the actual queue size is below sector five and the average queue size still below sector two, we duplicate the max_p parameter and use the accumulative drop probability p_a rather than using the current drop probability P_b to drop the arriving packets. In this scenario, p_a is the best drop probability because its value is higher than P_b. This helps to shrink the queue quickly.

Table 2: Parameter configuration for ERED and SQRED

	ERED	QSRED
min_{th}	*20*	*15*
max_{th}	*40*	*45*
min'_{th}	*50*	*30*
max'_{th}	*80*	*60*
P_b	$0.25\, max_p$	$0.67\, max_p$
Pa	*0.0025*	*0.00745*

7. Network Topology

Fig. 2 illustrates the network topology used to test QSRED with four network performance parameters. In this topology, six sources send packets with a maximum window size equal to *2000 Bytes*. The TCP version used here is TCP-Reno. A sink immediately sends an acknowledgment packet when it receives a data packet. Arrivals of sessions follow a Poisson process.

A connection between each node and the gateway has *1ms* delay time. The bottleneck link between the gateway and the sink has a *1ms* delay time for delivering the packet to the sink. Exponential distribution is used for the start time of packets transfers.

The following section describes our simulation results over the three introduced strategies with respect to four network performance parameters.

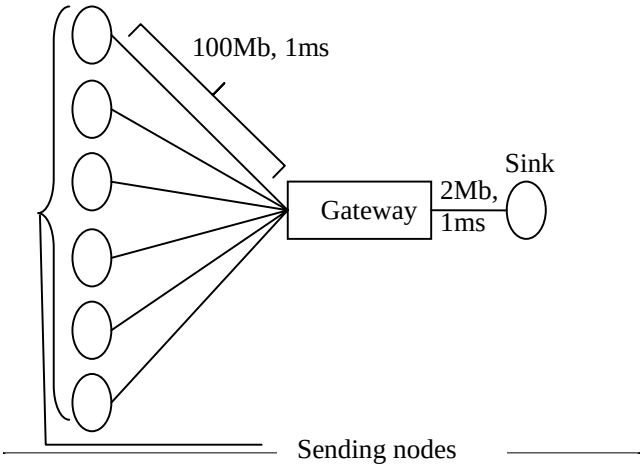

Fig. 2 The simulator network topology

8. Simulation Results

Figures 3 to 6, depict four network performance parameters for RED, ERED and QSRED strategies of congestion handling. These four parameters are the network throughput, link utilization, packets loss rate and average delay time. In addition, Fig. 7 shows network jitters. The simulation results are performed using the NS2 simulator [22].

QSRED shows better throughput, which is indicative of higher performance over the other two strategies. Additionally, QSRED shows a higher level of stability over RED and ERED. In the same manner, QSRED continued to reduce the packets loss rate when RED and ERED curves depicted higher loss rates.

With respect to the average delay time parameter, there were a few intersections in the curves of the three strategies. Despite these intersections, QSRED had a conservative delay time curve and it achieved lower delay time values during the simulation. This has been reflected in Fig. 7. QSRED appears the most stable whereas RED and ERED results show greater variance.

Fig. 3 shows the throughput parameter for RED, ERED and QSRED in bytes per second (Bps). The QSRED strategy has shown the best throughput values. It is clear that RED achieved the lowest throughput values, particularly at the end of the simulation when throughputs reached 207,000 Bps. It was observed that ERED throughput is higher than RED but it is still smaller than QSRED.

Fig. 4 plots the link utilization parameter. It demonstrates that QSRED link utilization is almost stable with the number of connections. In contrast, ERED and RED link utilization fluctuate dramatically. It is also noticeable that QSRED shows the highest link utilizations with variant number of connections.

Another important performance parameter, demonstrated in Fig. 5, is the packet loss rate. QSRED reduced the packet loss rate to 4% of the total number of packets propagated at time 30s. For RED, the loss rate was very high and reached 30% of the total propagated packets at the end of the simulation.

Fig. 6 illustrates the average network delay. Despite the oscillations with increasing number of connections, the results suggest that QSRED adds the lowest amount of delay. This suggests that QSRED maintains shorter average queue sizes. The figure shows that the maximum delay of RED exceeds 0.019s. For ERED this value exceeds 0.013s but with QSRED this value did not exceed 0.007s. Furthermore, QSRED delay time has been kept below 0.001s for the majority of simulation time.

Fig. 7 shows the amount of jitter introduced by the AQM strategies. Although the results appear similar, QSRED expressed the best network jitter results. The results, show that the minimum value for QSRED approaches -0.35ms and the maximum value is close to 0.13ms. Conversely, the values for ERED and RED ranged between -0.7ms to 0.33ms and -0.31ms to 0.33ms respectively. A qualitative analysis shows that the QSRED curve is the least erratic.

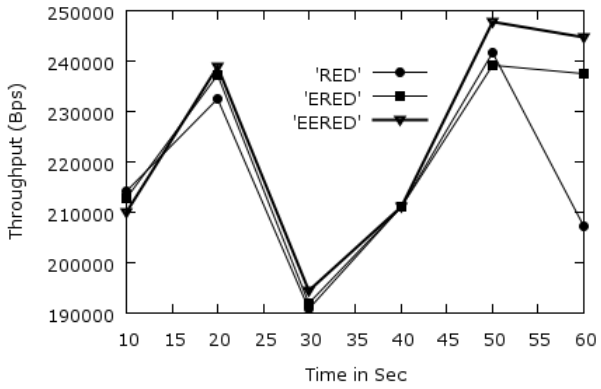

Fig. 3 Network throughput

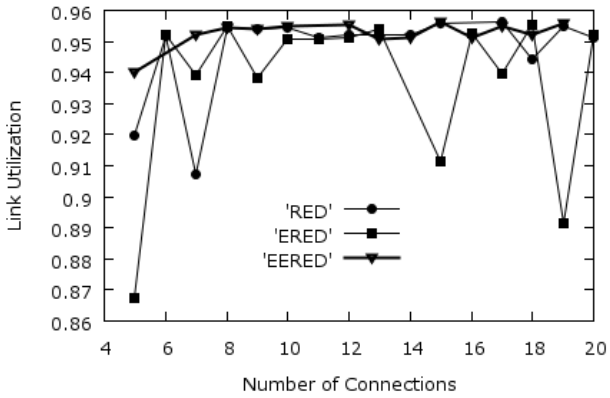

Fig. 4 Network link utilization

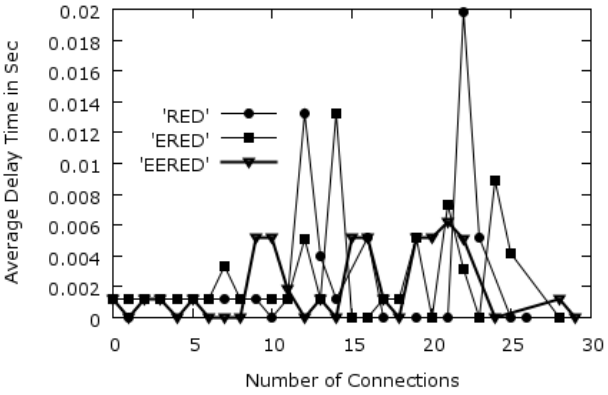

Fig. 5 Average network delay

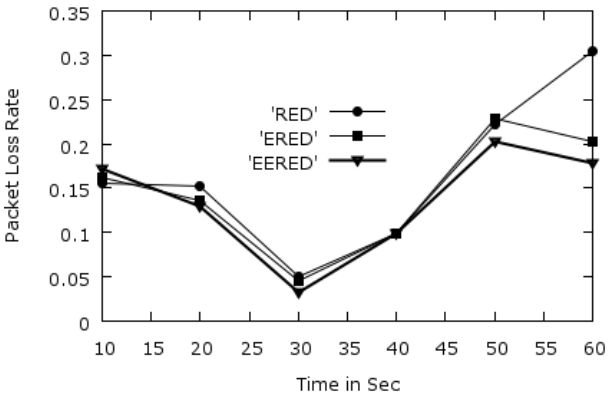

Fig. 6 Packet loss rate

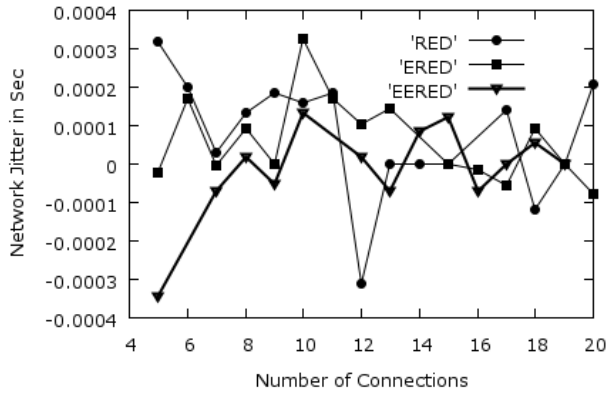

Fig. 7 Network jitter

9. Conclusions

This work introduces an efficient way to overcome the mismatch between the microscopic and macroscopic behaviors of queue length dynamics in RED gateways. We propose a new algorithm that enhances the implementation of the original RED strategy. RED has difficulties detecting this mismatch. This can result in RED continuing to drop packets after the congestion event has subsided. This work provides a new technique that helps the gateway manage network congestion and increase TCP performance.

Our algorithm, Queue Sectors RED (QSRED), divides the buffer into six equal sectors. These sectors represent new dropping levels added to the original RED implementation. It used the actual and the average queue size parameters to help RED absorb short lived bursty traffic and control TCP congestion efficiently.

Since RED uses probabilistic packet dropping, QSRED dynamically adjusts the max_p value of RED to maintain network stability and smooth traffic. We compared QSRED with ERED and RED strategies in NS2. The results show that QSRED offers higher throughput and link utilization with lower packet loss and lower delays.

References

[1] E. S. Hashem, "Analysis of Random Drop for Gateway Congestion Control," Massachusetts Institute of Technology1989.

[2] V. Jacobson, "Congestion avoidance and control," *SIGCOMM Comput. Commun. Rev.*, vol. 25, pp. 157-187, 1995.

[3] V. Firoiv and M. Borden, "A Study of Active Queue Management for Congestion Control," *IEEE INFOCOM*, 2000.

[4] S. Floyd. (2002, *RED (Random Early Detection) Queue Management*. Available: http://www.icir.org/floyd/red.html

[5] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Netw.*, vol. 1, pp. 397-413, 1993.

[6] S. Ryu, *et al.*, "Advances in Internet Congestion Control," *IEEE Communications Surveys and Tutorials*, vol. 5, 2003.

[7] T. Sheldon and B. Sur, "Congestion Control Mechanisms," 2005.

[8] V. Dumas, *et al.*, "A Markovian Analysis of Additive-increase Multiplicative-decrease algorithms " *JSTOR*, vol. 34, pp. 85-111, 2002.

[9] V. Jacobson, "Congestion Avoidance and Control," in *ACM SIGCOMM* 1988, pp. 314-29.

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

[11] W. Feng, "BLUE: A New Class of Active Queue Management Algorithms," *University of Michigan*, 1999.

[12] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," *IEEE/ACM Trans. Netw.*, vol. 8, pp. 556-567, 2000.

[13] S. Floyd, "Congestion Control Principles," *IETF RFC2309*, 1999.

[14] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the Internet," *IEEE/ACM Trans. Netw.*, vol. 7, pp. 458-472, 1999.

[15] M. May , *et al.*, "Reasons Not to Deploy RED," in *IEEE/IFIP IWQoS*, London , UK 1999, pp. 260-262.

[16] W. Feng, *et al.*, "A Self-Configuring RED Gateway."

[17] W. Feng, *et al.*, "The BLUE Active Queue Management Algorithms," *IEEE/ACM Transactions on Networking (TON)*, vol. 10, pp. 513-528, 2002.

[18] B. Abbasov and S. Korukouglu, "Effective RED: An Algorithm to Improve RED's Performance by Reducing Packets Loss Rate," *Journal of Network and Computer Applications*, vol. 32, pp. 703-709, May 2009.

[19] M. Christiansen, "Tuning RED for Web Traffic," *IEEE/ACM Trans. Netw.*, vol. 9, pp. 249-64, 2001.

[20] M. May, "Influence of Active Queue Parameters on Aggregate Traffic Performance," INRIA, Sophia Antipolis, France, 2000.

[21] T. J. Ott, *et al.*, "SRED: Stabilized RED," in *IEEE INFOCOM '99*, 1999.

[22] ISI. *The Network Simulator - ns-2*. Available: http://www.isi.edu/nsnam/ns/

Nabhan Hamadneh received the BSc. and MSc. degrees, from Irbid Uni. and Albalqa'a Applied Uni. in 2003 and 2005, respectively. He worked as IT teacher in the period 2004-2008. Since 2008, he is a full time PhD student (school of IT, Murdoch Uni.). His research interest is in networking especially TCP congestion control strategies and he had some of his work published in this area.

David Murray studied Inter-networking and Security at Murdoch University where he received his BSc in 2005. In 2007 he achieved a first class Honours in Computer Science. He has recently submitted his PhD in the area of wireless multi-hop ad-hoc networks. He has published numerous papers in this area. He is currently a lecturer at Murdoch University. His research interests include multi hop wireless architectures, cloud computing, routing and congestion control.

Michael Dixon received the BSc, "management information system" from California State Uni. in 1987, MBA Golden Gate Uni. in 1989 and PhD, Murdoch Uni. in 1999. He is a senior lecturer in telecommunications management at Murdoch Uni. His research interests include wireless communications, mobile computing, ad-hoc networks and congestion control in TCP networks.

Peter Cole is an associate professor and the Dean of the School of Information Technology, Murdoch University. He began his association with Murdoch University 23 years ago as a student and has been teaching in the IT faculty at Murdoch for over 17 years. Peter is the President of the Australian Council of Deans of ICT, a fellow of the Australian Computer Society and is actively involved on many industry initiatives both locally and nationally. Peter has been heavily involved with the development of international programs in Information Technology at Murdoch University.