IOSR Journal of Computer Engineering (IOSRJCE)
ISSN: 2278-0661 Volume 2, Issue 6 (July-Aug. 2012), PP 20-24
www.iosrjournals.org

Survey on Random Early Detection Mechanism and Its Variants

Alshimaa H. Ismail[1], Zeiad Elsagheer[2], I. Z. Morsi[3]
[1, 2](Dept. of Computer Science and Engineering, Faculty of Electronic Engineering/ Menofiya University, Egypt)
3(Dept. of Electrical Engineering, Faculty of Engineering / Menofiya University, Egypt)

Abstract: *Active Queue Management (AQM) algorithms are an Implementing schemes, So that packets are transmitted with higher priority than others. Random Early Detection (RED) is the first active queue management algorithm proposed for deployment in TCP/IP networks. RED has some parameter tuning issues that need to be carefully addressed for it to give good performance under different network scenarios. Various algorithms come from RED such as Stabilized RED (SRED), Dynamic RED (DRED), Adaptive RED (ARED) and Flow RED (FRED) these algorithms control congestion by discarding packets with a load dependent probability whenever a queue in the network appear to be congested, this paper will introduced some features about RED and its variants.*

Keywords: *Active Queue Management, Congestion control, Queue size, RED, TCP/IP*

I. Introduction

Congestion control is carried out in the transport layer at the source (end System) and has two parts, the first is end-to-end protocol (TCP/IP) which let senders adjust its sending rate according to the probability of packets being dropped in the network, the second part called active queue management (AQM) which implemented in routers [1, 2, 3, 4] As we mention AQM has many algorithms the most widely algorithm is Random Early Detection (RED). It's appeared in mid 80 by Sally floyed and Jacobson [1].

RED is tailored for TCP connection across IP routers it's designed to avoid congestion, global synchronization, and avoidance of bias against traffic and bound on average queue length to limit delay. RED is a queue length that marks packets with probability proportional to the current average queue length, for each arriving packet the average queue size is calculated using the Exponential Weight Moving Average (EWMA) [3, 4, 5, and 6]. The average queue size is compared with the minimum threshold and maximum threshold to determine next action, at packet arrival if average queue size is less than minimum threshold packet enter the queue but if average queue size is larger than maximum threshold packet dropped with a drop probability where a function of measured queue length [7, 8] is.

RED has its own variants which tend to control average queuing delay, while still maintaining high link utilization, reducing packet drops, reducing global synchronization and burst connection [3]. RED variants are an Implementing schemes, So that packets are transmitted with higher priority than others.

This paper is organized as follow; Section 2 gives details about RED variants. Section 3 gives a comparison about these variants. Section 4 the BLUE algorithm. Section 5 conclusions.

II. RED Variants

RED has many variants to overcome its drawbacks (i.e., RED doesn't store any state information about any flow). All variants depend on RED parameters (i.e., the average queue size, the minimum and maximum threshold) in dealing with congestion and achieving the highest QOS for router queue such:

Adaptive RED (ARED)

ARED Reduce both packet loss and the variance in queuing delay by minimizes the possibility of overshooting (i.e., keeping the average queue size within target range half way between minimum threshold and maximum threshold [3, 9], It will work to not go underneath a packet loss probability of 1% and it will not exceed a packet loss probability of 50%. This is done to maintain acceptable performance even during a transient period where the average queue length moves to the target zone. ARED can be expressed as the following as we see at [4, 10].

Table.1

```
Every Q(avg) update:
if ( MINth <   Q(avg) < MAXth )
Status = Between;
if ( Q(avg) < MINth && status != Below)
Status = Below;
maxp = maxp/α ;
if ( Q(avg) < MAXth && status != Above)
status = Above ;
maxp = maxp * β ;
```

Stabilized RED (SRED)

SRED is designed to stabilize the queue size at a level independent of the number of active connections, the packet drop probability are computed by estimating the active flows to adjust instance queue size [13, 9].SRED maintains a virtual list viewed as a container called a zombie list which stored source and destination address for each arrival packet. When the list is full a random zombie picked up from a list and compared with the source and destination addresses for the new packet. If there is a match (i.e., belong to the same flow) Hit (i.e., declared to calculate the traffic load) is set to one. Otherwise, it set to zero, and with a certain probability P, the content of this zombie may be replaced by the source and destination of this new packet [8, 9]. The Hit frequency P(t) will be updated after each packet arrival, and it can be estimated by using an exponentially weighted moving average filter [13] expressed by the following equation.

$$P(t) = (1-\alpha)P(t-1) + \alpha Hit(t) \quad , (1)$$

Where $\alpha \approx p/M$, p is the probability of updating the zombie list when Hit is zero, and M is number of the zombies in the list. It was shown that P(t)-1 is a good estimate of the number of active connections [8, 13].

SRED algorithm can stabilize the queue size at a level independent of the number of active connections. The basic drop probability Psred of SRED is related to the queue size as follow [13].

$$P_{sred} = \begin{cases} P_{\max} & \text{if } B/3 \le q < B \\ P_{\max}/4 & \text{if } B/6 \le q < B/3 \\ 0 & \text{if } q < B/6 \end{cases} \quad , (2)$$

Where Pmax = 0.15. The full SRED drop probability P_{zap} is associated with the frequency of Hit, basic drop probability and Hit value [8].

$$P_{zap} = P_{sred} \times \min\left(1, \frac{1}{(256 \times P(t))^2}\right) \times \left(1 + \frac{Hit(t)}{P(t)}\right) \quad , (3)$$

Fig.1 show the effect of SRED in the queue size as shown in the original paper [13] for 1000 TCP connection, Number of the zombie in the zombie list is 1000, The maximum drop probability (Pmax) is 0.15, The refresh probability to update the zombie list (P) 0.25, Buffer Size (B) 860 packets :

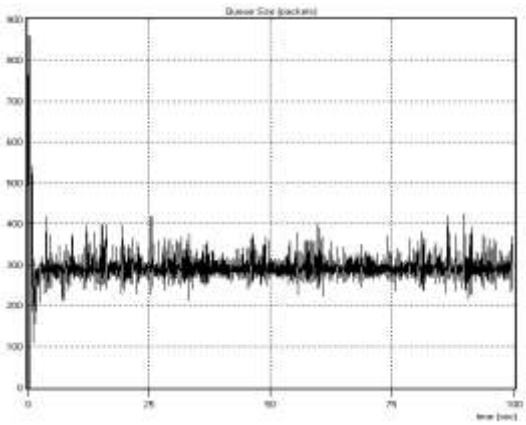

fig.1: SRED effectively controls the queue size at the expense of a larger buffer size

Flow RED (FRED)

RED only keeps track of flows that have packets in the buffer, so FRED penalizing non adaptive flow by imposing a maximum number of buffered packets; it based on calculating average queue length at both packet arrival and departure to avoid RED drawbacks [2, 8]. FRED Protecting fragile flows by deterministically accepting flows from low bandwidth connections, also providing fair sharing for large numbers of flows by accepting two- packet-buffer [4, 5, 11].

FRED has two parameters (MINq) and (MAXq) which are the minimum and maximum numbers of packets that each flow is allow to buffer. FRED uses a global variable avgcq to estimate the average per-active-flow buffer usage. It maintains the number of active flows and for each of them, FRED maintains a count of buffer packets, Qlen, and a count of times when the flow is responsive (Qlen>MAXq).FRED will penalize flow with high strike values [5, 8 and 11].

Dynamic RED (DRED)

DRED stabilize the queue size while keeping high link utilization and controlling the packet loss rate. DRED is a simple feedback control approach to randomly discard packets [8, 9]. Probability of dropping packet of DRED is responsive enough to traffic and as a result there will not be buffer overflow at the gateway, the next figure show the probability of dropping packets as a function of average queue for DRED.

The operation of DRED is simple and can also be specified as a sequence of steps, carried out at time n. First, the the current queue length Q(n) is sampled. Then, the current error signal E(n) is computed as E(n) = Q(n) − Qref where qref is the target queue ,with the difference that no low-pass filter has been applied yet to the observed queue length, meaning Q(n) represents the instantaneous queue length [7, 9 and 12]. To this error signal e(n), a low-pass filter is then applied using

$$\hat{e}(n) = (1-\beta)\,\hat{e}(n-1) + \beta\, e(n) \quad , (4)$$

Where β is again the preset low-pass filter gain. The dropping probability [8] P(n) can now be computed from equation (5)

$$p(n) = \begin{cases} 1 & \text{if } p(n-1) + \alpha \dfrac{\hat{e}(n)}{B} > 1 \\ p(n-1) + \alpha \dfrac{\hat{e}(n)}{B} & \text{if } 0 \le p(n-1) + \alpha \dfrac{\hat{e}(n)}{B} \le 1 \\ 0 & \text{if } p(n-1) + \alpha \dfrac{\hat{e}(n)}{B} < 0 \end{cases} \quad , (5)$$

where α is the control gain that preset maximum dropping probability, and β is the buffer size . The dropping probability P(n) is then stored for use at time N+1, when a new probability P(N + 1) will be computed using the steps defined above. For this purpose, $\hat{e}(n)$ needs to be stored as well. Simulations have shown that DRED is indeed able to stabilize the router queue length close to a predefined reference value, and accommodates well to traffic bursts without dropping too many incoming packets [4, 5 and 8]. For the popular evaluation [8] Sampling Interval (Δt)10 packet transmission time, Control Gain (α)0.00005, Filter Gain (β)0.002, Control Target (T) 293 packets, Buffer Size (B) 586 packets, No-drop Threshold (L) 264 packets, Initial Drop Probability 0.05. fig.2 is a result of applying the DRED algorithm with the pervious parameters which show that DRED can stabilize its queue size around the control target (T) (which is set between the minimum and maximum threshold of RED).

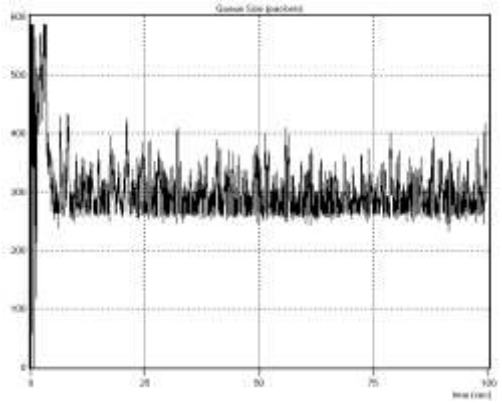

fig.2: DRED maintains the queue size around the control target

III. The BLUE Algorithm

BLUE [13] is an Active Queue Management algorithm. Like RED, it operates by dropping or marking packets in a router's queue before it overflow. BLUE maintains a single probability Pm, to mark (or drop) packets. If the queue is overflow, BLUE increases Pm, thus increasing the rate at which it sends back congestion notification or dropping packets [5, 12, 14]. Conversely, if the queue becomes empty BLUE decreases its marking probability. Marking probability Pm is also updated when the queue length exceeds a certain value in order to allow room to be left for transient bursts and to control the queuing delay when the size of the buffer being used is large. The basic blue algorithms can be summarized as following [4, 14]:

Table.2

Upon Packet loss event: if ((now – last_update) > freeze_time) pm := pm + / d1 last_update := now	Upon link idle event: if ((now – last_update) > freeze_time) pm := pm - / d2 last_update := now

If we set the parameters [8] Initial Drop Probability 0.05, Freeze time period 0.01, Increase drop probability (d1) 0.00025, Decrease drop probability (d2) 0.000025, Buffer Size (B) 450 packets [9], It's obvious that the BLUE algorithm suffers from queue size oscillations after applying the previous parameters as shown in fig.(3)

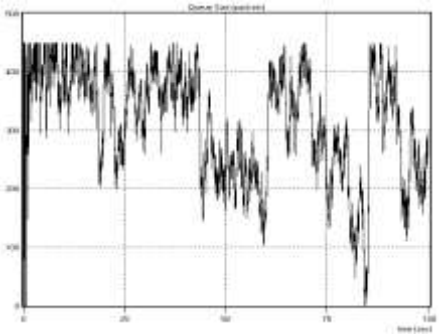

fig.3: the BLUE algorithm suffers from queue size oscillations

The differences between BLUE, DRED and SRED queue size are shown in fig.(4).

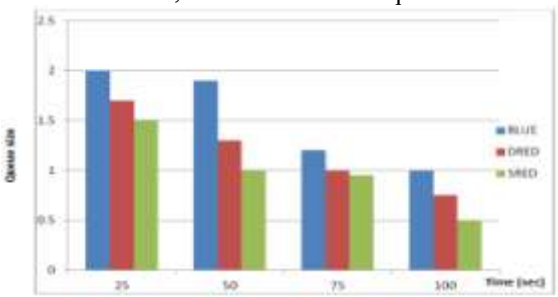

fig.4: differences between BLUE, DRED and SRED queue size

The pervious chart show that the SRED stabilize the queue size very well than other algorithms

IV. Conclusions and future work

RED doesn't able to stabilized the queue size, while DRED and SRED both stabilize the queue size very well and also they have more predictable packet delay inside the network. SRED has higher drop probability and higher packet loss rate than DRED. RED, ARED and DRED monitors the queue length while SRED can monitor the queue length and packet header, RED has a hard time maintaining the queue size between the two thresholds. SRED effectively controls the queue size at the expense of a larger buffer size. DRED is continuously adjusting its drop probability to reflect any change in the traffic load, Compared to DRED [4, 8, 12].

For future work as we see that to overcome the congestion in the router queue we concentrate on three metrics (i.e., queue size, packet loss, and the dropping probability) to control the network congestion in an

efficient way and improve QOS. We will work to deal with the three metrics to get best effort services and we can also work to decrease the queue delay as it considers another metric.

References:

[1] S.Floyed, V.Jacobson, Random Early Detection gateways for congestion avoidance, IEEE/ACM Transactions on Networking August 1993.
[2] Teresa Álvarez, Virginia Álvarez, Lourdes Nicolás, UNDERSTANDING CONGESTION CONTROL ALGORITHMS IN TCP USING OPNET, Spain, 2010.
[3] Jianyong Chen, Cunying Hu, and Zhen Ji , Self-Tuning Random Early Detection Algorithm to Improve Performance of Network Transmission, Hindawi Publishing Corporation Mathematical Problems in Engineering Volume, Article ID 872347, 17 pages doi:10.1155/2011/87234, 2011.
[4] IMPROVING INTERNET CONGESTION CONTROL AND QUEUE MANAGEMENT ALGORITHMS by (Wu-chang Feng), A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, (Computer Science and Engineering) in The University of Michigan 1999.
[5] G.F.Ali Ahmed, Reshma Banu, Analyzing the performance of Active Queue Management Algorithms, International journal of Computer Networks & Communications (IJCNC), Vol.2, No.2, March 2010.
[6] Chandra Himanshu, Agarwal Ajay, Velmurugan T, Analysis of Active Queue Management Algorithms & Their Implementation for TCP/IP Networks Using OPNET Simulation Tool, International Journal of Computer Applications.
[7] S.Dijkstra, Modeling Active Queue Management algorithms using stochastic Petri Nets, Faculty of Electrical Engineering, Mathematical and Computer Science, University of Twente, December 14, 2004.
[8] Chengyu Zhu, O.W.W. Yang, J. Aweya, M. Ouellette, Montuno, A comparison of active queue management algorithms using the OPNET Modeler, Communications Magazine, IEEE, volume 40, Pages: 158 – 16, June 2002.
[9] Michael Welzl, Leopold Franzens Network Congestion Control Managing Internet Traffic, University of Innsbruck.
[10] R. J. La, P. Ranjan, and E. H. Abed, Analysis of Adaptive Random Early Detection (ARED), Networking, IEEE/ACM Transaction, Volume 12, Pages: 1079 – 1092, 2004.
[11] G.Thiruchelvi and J.Raja, A Survey on Active Queue Management Mechanisms, IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.12, December 2008.
[12] Victor Firoiu, Marty Borden , A Study of Active Queue Management for Congestion Control, Nortel Networks, Billerica, MA, INFOCOM 2000, Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies Proceedings. IEEE, Volume 3, Pags:1435 - 1444 vol.3, Mar 2000 .
[13] Sunitha Burri, BLUE: Active Queue Management CS756 Project Report, May 5, 2004.
[14] Wu-chang Feng, Member, AQM, Kang G. Shin, Fellow, IEEE and ACM, Dilip D. Kandlur, Member, IEEE, Debanjan Saha, Member, IEEE, The BLUE Active Queue Management Algorithms, volume 10, Augest 2002.