A Survey in Active Queue Management Methods According to Performance Measures

Mahmoud baklizi[1], Jafar Ababneh[2]

[1]*Department of Computer Information and Network Systems (CINS), The World Islamic Sciences & Education University (W.I.S.E), Amman, Jordan*

Abstract *Congestion generally occurs when the amount of packets arriving at the router buffer exceeds the available resources. This causes several problems, such as, increase in the probability of high queuing delay in the buffer, and increase in the probability of losing packets of the buffer. Congestion control method is one of the key that keeps any network efficient and reliable for the users. Many researchers were proposed in the literature over theses years for the efficient control of congestion that occur in the network. The congestion is monitored and controlled at an early stage before the router overflows, using a set of parameters. These methods are formally referred to as Active Queue Management (AQM) methods, which were proposed to overcome the congestion. These methods depend on maintaining the router buffer dynamic. When the packets arrival increases, the amount of packets dropped increases to prevent the accumulation of packets and to maintain the stability of the buffer. This procedure is implemented by assigning a fixed value at which the aql should be maintained. Packet dropping begins before the buffer overflows in reference to the fixed value and the current value of aql. Although such methods perform well in steady buffer, they do not adapt well when aql changes over time. In this paper, In this paper an comprehensive survey is made on the AQM methods that are proposed and the values and short tumbles is existing.*

Keywords *Congestion control, Active Queue Management, router buffer, Performance Measures.*

I. Introduction

This document describes, and is written to conform to, author guidelines for the journals of AIRCC series. The massive amount of applications connected via computer networks and Internet technologies generate huge packet transmissions among different parties. The transmission process between the sender and receiver is intermediate with the queuing process occurring in the network router buffers. Packets that arrive first at the buffer queue are processed and transmitted first, in First-In-First-Out (FIFO) manner. Subsequently, if the buffer size is large, arrived packets faces delay before they are departed, this decreases the performance of the router and the performance of the network as a whole. On the other hand, if the buffer size is small, the buffer will be overflowed quickly and the arriving packets will have no space to be accommodated, which causes packet loss as in [1].

aql denotes the average number of packets queued in the router's buffer at a point in time, Congestion generally occurs when the amount of packets arriving at the buffer cannot be accommodated because of insufficient *aql*, average queue length. Which is not suitable for the amount of packet transmissions occurring in the network? Consequently, congestion can be assessed and controlled by *aql*. Congestion elimination or prevention is required to optimize routing efficiency and network resources as well as enhance network performance as in [2].

Several pre-congestion methods, such as active queue management (AQM) [3], were proposed almost a decade ago to detect and prevent congestion at an early stage [4]. These methods depend on maintaining the *aql* value at medium level. When the packets arrival increases, the amount of packets dropped increases to prevent the accumulation of packets and to maintain the stability of the buffer. This procedure is implemented by assigning a fixed value at which the *aql* should be maintained. Packet dropping begins before the buffer overflows in reference to the fixed value and the current value of *aql*. Although such methods perform well in steady buffer, they do not adapt well when a*ql* changes over time.

Existing pre-congestion control methods generally do not adapt well to the changes in *aql* value. This causes several problems when *aql* changes over time. These problems can be summarized as (1) increase in the probability of high queuing delay in the buffer, and (2) increase in the probability of losing packets because of buffer overflow [1, 4]. As a result, these methods cannot stabilize *aql* well at an optimal level [5].

In summary, existing pre-congestion control methods do not adapt well when network traffic changes over time, leading to waste of network resources in general and increase in probable packet loss and average waiting time of packets in particular. Thus, an adaptive method capable of detecting congestion before the buffer router overflows must be established. In addition, a non-parameter solution should be adapted in order to overcome the parameterization problem

II. Congestion Control Methods

The following Congestion control methods are continuously linked with the rapid advances in Internet and network technology. Most of these methods have been built by filling up the gap in their formers. Fig. 1 gives an overview of the three categories of the congestion control methods, namely late congestion control method, parameter-based early congestion methods and Fuzzy-based early congestion methods

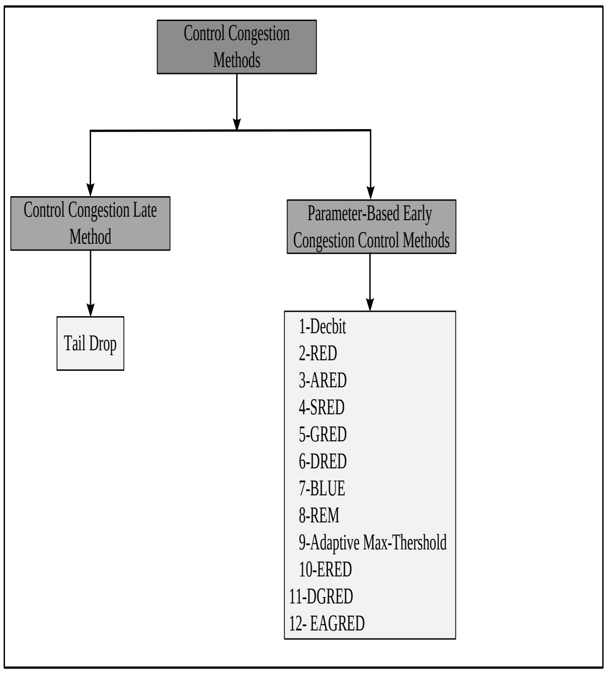

Figure 1. Classification of congestion control methods

The state-of-the-art methods in congestion control are discussed in this section. To show the evolving procedure, the discussion starts with the early methods that are based on post-congestion detection and then processed with the latest method that implements pre-congestion control.

A. Late Congestion Control Method

In the late congestion control category, congestion control sets up the size of the buffer in the router to a fix value these types are:

1) The Drop Tail (DT)

The Drop Tail (DT), a non-adaptive (the parameters values never changed after they have been initialized) method, controls congestion by fixing the size of the router buffer to optimize delay and packet loss [6]. Congestion is controlled, when the buffer overflows by dropping all incoming packets automatically. Generally, the size of the buffer maybe set to any value, intermediate, maximum or minimum size see Fig. 2 [7, 8].

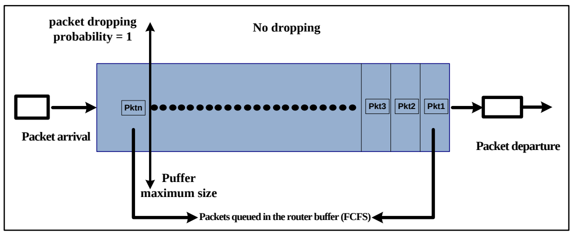

Figure 2. Router buffer control using DT

B. The Parameter-Based Early Congestion Control Methods

In parameter-based early congestion control methods, the congestion is monitored and controlled at an early stage before the router overflows, using a set of parameters. These methods are formally referred to as Active Queue Management (AQM) methods, which were proposed to overcome the limitations of the DT method discussed earlier [9, 10]. Enormous methods for congestion control have been built as AQM, such as Random Early Detection (RED) [11], Adaptive Random Early Detection (ARED) [12], Random Exponential Marking (REM) [13, 14] , BLUE [15, 16], Stochastic Fair BLUE (SFB) [17], Gentle Random Early Detiction (GRED) [18], Dynamic Random Early Drop (DRED) [19], Stabilised Random Early Drop (SRED) [20], DRED [21, 22] Fuzzy BLUE[23], Fuzzy Exponential Marking (FEM) [24], Decbit [25] and Adaptive Gentle Random Early Detection.

All of these methods are adapted to the congestion status at the router buffer by changing the values of the utilized parameters based on the network status.

1) Decbit Method

The Decbit [25], an adaptive (the parameters values are changed dynamically from situation to another situation) method, uses the average queue length (*aql*) as congestion indicators in the network. The In the case that DT method sets the router buffers to the maximum size, the possibility of high packet queuing delay D is raised. In the case that DT sets the router buffers to a minimum size, the throughput T decreases rationally. The advantage of DT is the simplicity in implementation with slight calculation overhead

The disadvantages of DT are the difficulty in obtaining an optimized value for all the performance measure, increase in packet loss rate and saturation of the queue router buffer [26].

Congestion is controlled by sending up a notification to the sender to reduce the transmission rate. As such, if the *aql* value, the ratio between the numbers of packets currently in the queue to the size of the queue, exceeds the value of one, then a notification is sent to the sender, via an indication bit, in order to decrease the transmission rate.

To clarify, for every packet that arrives at the router buffer, Decbit method calculates the current *aql*, if *aql* exceeds one, the decbit method marks the arriving packet and transfers it to its destination. Based on the indication bit, an acknowledgment from the destination is sent back to the source, and the source can identify the congestion from the acknowledgment's content and subsequently decreases the transmission rate to avoid congestion. The router buffer based on decbit method is illustrated in Fig. 3.

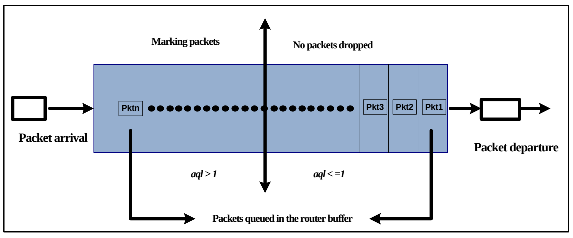

Figure 3. Router buffer control using Decbit

Usually, the sources modify their transmission windows once each two round trip times (RTTs), using Decbit method, if half or more of the packets in the last window size are marked by the congestion indication bit, then the transmission window-size decreases exponentially [13, 14]. On the other hand, the transmission rate increases linearly if no packets are marked [11, 12, 18].

In general, the advantages of decbit are being simple, distributed, optimized, low overhead congestion as the feedback is sent by marking packets and dynamic which provides good fairness. On the other hand, decbit is not suitable for bursty traffic as it takes time to notify the sender to reduce the transmission rate.

1) *Random Early Detection (RED)*

The Random Early Detection (RED), a non-adaptive method, was proposed by Floyd and Jacobson in 1993. RED does not notify the sender as decbit, it controls congestion using probabilistic packet dropping.

RED uses *aql* and two calculated thresholds values, namely, *minthreshold* (minimum threshold) and *maxthreshold* (maximum threshold) as congestion indicators (see Fig. 4.).

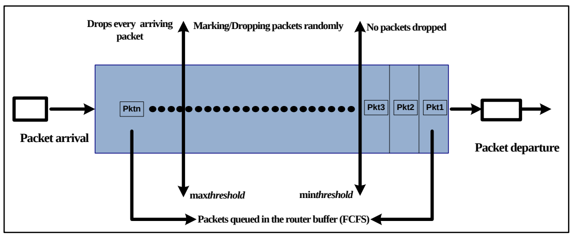
Drops every arriving packet
Marking/Dropping packets randomly
No packets dropped
Pktn
Pkt3
Pkt2
Pkt1
Packet arrival
Packet departure
maxthreshold
minthreshold
Packets queued in the router buffer (FCFS)

Figure 4. The single router buffer for RED

The congestion is controlled using various scenarios as follows: First, when *aql* is smaller than the *minthreshold*, no packets are dropped as congestion does not occur in this case. Second, if the *aql* is between the two thresholds, the arriving packet is dropped with calculated probability D_p to alleviate congestion before the buffer overflowed. Finally, when the *aql* is above the *maxthershold*, all arriving packets are dropped. By other means, the dropping probability D_p value is set to one (Figure 4).

RED is one of the most significant methods as it has been successfully adopted by the Internet Engineering Task Force (IETF) in RFC 2309 [6]. The advantage of RED is the elimination of the global synchronization problems. However, RED's drawback is that the congestion indicator is embodied in computing *aql* based on the traffic load (number of connections), not on the actual status of the packet load, which may degrade the network performance in many aspects. Such as, delay and packet loss. Subsequently, RED cannot stabilize its *aql* value between the *minthershold* and *maxthreshold* when the traffic load changes suddenly (i.e., bursty traffic) [12, 16].

3. *Adaptive Random Early Detection (ARED)*

The Adaptive Random Early Detection (ARED), an adaptive method, was proposed to overcome *aql* stabilizing problem that occurred in RED [27]. Although, ARED and RED have identical congestion indicator techniques, the difference between these methods is embodied in the congestion control technique. The ARED stabilizes *aql* at a specific value called T_{aql} as illustrated in fig. 5., which is computed between the *minthreshold* and the *maxthershold* [12]. Subsequently, the ARED method prevents the queue from growing up by preventing the *aql* value reaching the *maxthershold* [12].

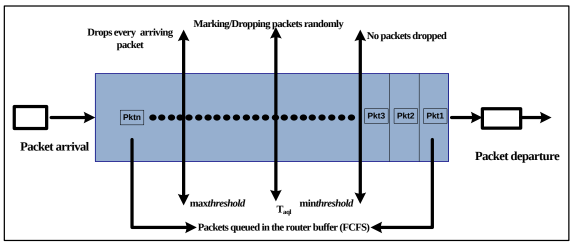
Drops every arriving packet
Marking/Dropping packets randomly
No packets dropped
Pktn
Pkt3
Pkt2
Pkt1
Packet arrival
Packet departure
maxthreshold
T_{aql}
minthreshold
Packets queued in the router buffer (FCFS)

Figure 5. The single router buffer for ARED

Besides T_{aql} a variable called Dmax, is given to stabilize the *aql* based on one of the following scenarios:

When aql at the router is below the T_{aql} and D_{max} is greater than or equal 0.01, D_{max} is decreased and subsequently the dropping probability is decreased.

When *aql* is above T_{aql} and D_{max} below or equal 0.5, D_{max} is increased and subsequently the dropping probability is increased.

The main advantages of ARED are eliminating, to some extent, the parameters sensitivity that affects RED's performance and the ability to stabilize *aql* partially. The ARED limitations are as follows. First, ARED cannot stabilize the *aql* between *minthreshold* and *maxthreshold* when heavy congestion occurs. Second, similar to RED, if a heavy congestion occurs in ARED while the aql is less than *minthershold*, then the router buffer is likely to be overflowing and losing all packets. Finally, ARED's parameters have to be set up to specific values in order to obtain a satisfactory performance (parameterization).

3. *Stabilized Random Early Drop (SRED)*

Stabilized Random Early Drop (SRED), an adaptive method, was proposed to overcome the dependency problem between the computed *aql* and the number of TCP connections found in RED [20]. SRED method identifies the connections that take more than their share of bandwidth and obtains fair share for all the active connections with no calculations overhead. Consequently, SRED uses zombie list to store the information of the recently active flows, such as, counting their variables and time stamps. See Fig. 6.

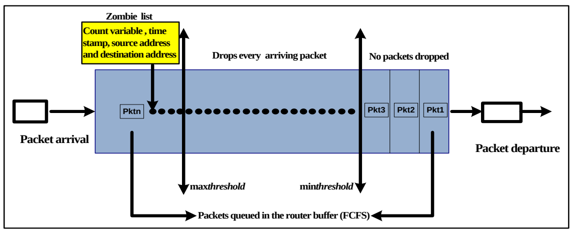
Zombie list
Count variable , time stamp, source address and destination address
Drops every arriving packet
No packets dropped
Pktn
Pkt3
Pkt2
Pkt1
Packet arrival
Packet departure
maxthreshold
minthreshold
Packets queued in the router buffer (FCFS)

Figure 6. The single router buffer for SRED

SRED operates as follows: SRED starts operating with empty zombie list. After that, as the packets arrive to the router buffer, the SRED identifies the source and destination addresses in the flow and store this information, with several other information in the zombie list. In case the zombie list gets full, every arrived packet is compared with randomly drawn record from the zombie list. If the compared records matched, called hit in this case, the matched zombie record repetition is increased by one and time stamp of this record is updated to match the time stamp of the latest packet arrival time. Yet, if the flow of the arrived packets is not in the zombie list (no hit), with a probability, the underlying flow information overwrites a zombie record and the repetition counter of that record is set to zero [20].

Generally, when many arrival packets match several zombie records, many hits are produced and the counts are incremented many times accordingly. Now, from the definition of misbehaving flows (flows which have gained more than their fair share in bandwidth), flows with high count values and several hits are those that are likely to be misbehaving.

As the records are filled, SRED estimate the frequency of packets hits P(m) = N/D, N denotes the size of the zombie list where D denotes the probability that two packets are not a match. Then, the number of active flow is computed by taking the inverse of P(m) value, i.e (P(m)-1). [20]. Subsequently, D_p is calculated according to P(m)-1, ql and D_{max}, based on the following scenarios:

When *ql* is below the K/6 , location at the SRED router buffer. where K represents the buffer capacity, no packets are dropped,

When *ql* is between K/6 and K/3 then D_{max} sets to D_{max} /4. Subsequently D_p is increased.

Finally, if the *ql* reach K/3 , the D_{max} still D_{max}, and the D_p is increased subsequently.

when 1>= P(m)>= 1/2566 , D_p is increased, the selection of 2566 has made stochastically, and needs further investigation [20].

The advantage of SRED is degrading the dependency of *aql* by using the zombie list. On the other hand, the SRED's drawback is embodied in high packet loss. SRED chooses to keep the queue short and causes loss rate to increase with the incremental of the number of connections [28].

3. *Gentle Random Early Detection (GRED)*

Gentle Random Early Detection (GRED), an adaptive method, was proposed by Floyd to overcome the limitations of RED [18]. GRED uses the same congestion indicator technique as RED. GRED stabilizes the aql at a certain level using three thresholds, namely, *minthershold*, *maxthershold*, and *doublemaxthershold*.

GRED uses another threshold value called *doublemaxthershold* and introduces different probabilistic dropping rates between these thresholds. This makes GRED better than RED in stabilizing the aql because when the aql exceeds the *maxther`shold* a higher probability is used to prevent buffer overflow.

while the stabilization mechanisms of GRED and RED are different, calculating D_p in GRED is partially similar to the one in RED. Generally, GRED reacts with the arriving packets based on one of the following scenarios (Fig. 7).

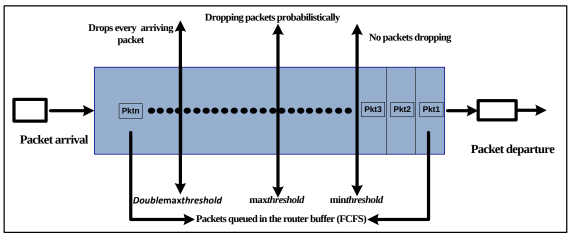

Figure 7. The single router buffer for GRED.

When *aql* at the router is below the *minthreshold*, no packets are dropped.

When aql is between the *minthreshold* and *maxthreshold*, the router will drop the arriving packets randomly, similar to RED.

When *aql* is between the *maxthreshold* and the *doublemaxthreshold*, the packets are dropped randomly with a higher probability compared to the previous case.

When *aql* is equal or greater than the *doublemaxthreshold*, the GRED router drops the arriving packets with Dp equal to one (i.e., arriving packets are dropped).

Unfortunately, GRED has some limitations. First, GRED deals with several threshold values and must set its parameters to specific values to obtain satisfactory performance (i.e., parameterization). Second, when the aql is less than the *minthreshold* and heavy congestion occurs, the *aql* will take time to adjust, during which the router buffer will likely overflow. Thus, no packets are dropped despite the overflowing GRED of router buffer [12, 18, 19].

3. *Dynamic Random Early Drop (DRED)*

Dynamic Random Early Drop (DRED), an adaptive method, was proposed by [19] to overcome the *aql* dependency [6]. Unlike, GRED, which stabilizes the aql by using three thresholds, with no target *aql* value, the main goal of the DRED method is to stabilize the aql at defined level (Tql) regardless of the number of connections in the network and using a single threshold for one node as shown in Fig. 8.

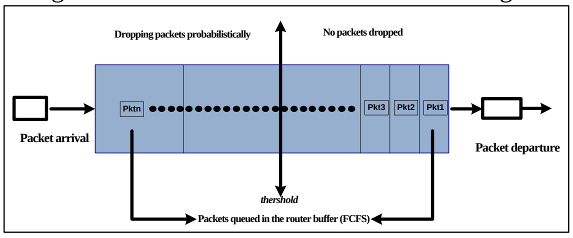
Dropping packets probabilistically
No packets dropped
Packet arrival
Pktn
Pkt3
Pkt2
Pkt1
Packet departure
thershold
Packets queued in the router buffer (FCFS)

Figure 8. The single router buffer for DRED

The Dropping probability D_p, is calculated in each time unit, based on the queue length, *ql*, in that specific time unit. A variable for dropping threshold (*th*) is used to achieve high utilization. Therefore, no updates for D_p and no packet dropping occur when $ql < th$, updates is taking place only if $ql > th$. This indicates that *ql* is a crucial parameter for detecting the congestion in DRED [19]. Finally, DRED marks packets by adding an ECN bit[29] in their headers or drops packets at its router buffers.

The main limitations of DRED are as follows: First, the DRED has to set its parameters to specific values to provide a good performance (parameterization). Second, DRED uses the instantaneous queue length rather than the average queue length which may result in unnecessary probabilistic dropping of packets when the congestion is of a very transient nature.

3. *Blue Method*

BLUE method, an adaptive method and similar to ARED, GRED and DRED, was proposed to enhance the performance of the RED method [15, 16]. BLUE relies on a DP and a certain threshold, similar to DRED method. When the buffer length at the router is larger than the threshold, BLUE increases D_P to manage the congestion (see fig. 9).

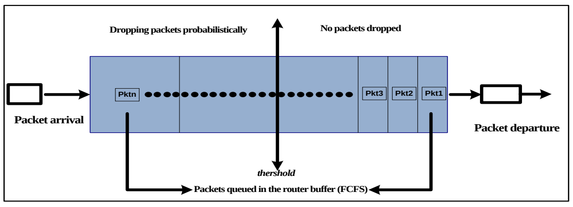
Dropping packets probabilistically
No packets dropped
Packet arrival
Pktn
Pkt3
Pkt2
Pkt1
Packet departure
thershold
Packets queued in the router buffer (FCFS)

Figure 9. The single router buffer for DRED

Further, if the buffer length is empty or the link is idle, then DP will be decreased. Unlike RED, which utilizes the *aql* as a main congestion metric, [11] , BLUE relies on the packet loss, the link utilization and the buffer length at the router as congestion indicators [15, 16]. The packet loss and the buffer length are tuned by adjusting DP, whereas, the link utilization is achieved through the link status. BLUE avoids the problems associated with bursty traffic flows by allowing space to accommodate the bursty packets in its router buffer by stabilizing the value of *aql* at a target level T_{aql}.

The limitations of BLUE method are the same as DRED method: First, BLUE required setting up its parameters to specific values to provide a good performance (parameterization). Second, BLUE uses the instantaneous queue length rather than the average queue length which may result in unnecessary probabilistic dropping of packets and increase queuing delay when the congestion is of a very transient nature.

3. *Random Exponential Marking (REM) Method*

Random Exponential Marking (REM), an adaptive method, aims at enhancing packet loss and queuing delay [13, 14]. REM has two main properties, transmission rate (R) and queue length (*ql*) comparison property and the total price property. The aim for the first is to stabilize the transmission rates (R) of the sources at the link capacity (L) and stabilize the (ql) at a certain level (target queue length (T_{ql})). The price property is used to find out the DP which is used as congestion metric. The price property depends on the rate mismatch and the queue mismatch, where the rate mismatch corresponds to the difference between the sources transmission rates and the link capacity, and the queue mismatch equals the difference between the queue length and the target queue length [13, 14].

When the source sends a particular packet through n routers' buffers along the path to its destination, each router buffer calculates its price value, and the total prices for the router buffers are used in calculating the DP along the path from source to its destination. When some router buffers at the path between source and destination are congested, their price values increase, and this explains the exponential growth in the D_P. In order to alleviate the congestion, the routers inform the sources about the congestion, and the sources adjust their transmission rates. REM method stabilizes rate around the link capacity and also stabilizes the queue around a small target. Unfortunately, REM has limitations such as, parameterisation and low throughput when the traffic is high.

3. *Adaptive Maximum Threshold*

Adaptive Maximum Threshold [30], an adaptive congestion control method, uses *aql* as congestion indicator. Thus, Adaptive Maximum Threshold method controls congestion in a similar way that has been used in RED with only one variance embodied in using an adaptive *maxthershold* position. This method also aims at stabilizing the *aql* value at T_{aql} (see Fig. 10.), which is the middle value between minimum and maximum threshold [30]. Unlike RED, this method uses extra *maxthershold*, called maxthreshold2. Like GRED, this method uses two dropping probabilities to stabilize *aql*.

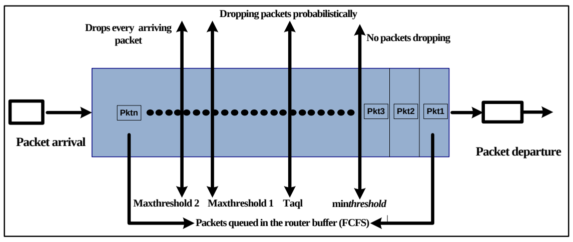

Figure 10. The single router buffer for An Adaptive Maximum Threshold

Stabilizing *aql* at T_{aql} prevents the router buffer from building up, and thus the packet loss prevented. An Adaptive Maximum Threshold calculates the packet dropping probabilities based on the *aql* position see figure 10 [30].

The congestion is controlled using various scenarios as follows:

When *aql* is smaller than the *minthreshold*, no packets are dropped.

When, in this case, aql is between the *minthreshold* and maxthreshold1, D_p is increased.

Subsequently, when *aql* is in the case, between the maxthreshold1 and maxthreshold2, D_p is increased accordingly.

When the *aql* is above the maxthershold1, all arriving packets are dropped. By other means, the dropping probability D_p value is set to one.

Adaptive Maximum Threshold is stabilized the *aql* between *minthershold* and *maxthershold* partially. Adaptive Maximum Threshold drawback is its parameterization and it has many *thersholds*. Also, when the traffic is high, this method takes time to adjust.

3. *Effective RED (ERED) Method*

The Effective RED (ERED) [2], an adaptive method, was proposed by Babek and Serdarin 2009 , to overcome some of RED's drawbacks and limitations [2]. ERED uses the instantaneous queue size as congestion indicator. ERED changes the values of *minthershold* and *maxthershold* parameters adaptively. Like GRED, ERED uses several thresholds to set up different probability for packet dropping based on the congestion status.

When the packets queue grows, the value of *maxthershold* is set to equal double of *maxthershold* and *minthershold* equal *minthershold* plus *maxthershold* over two plus *minthershold*. Congestion is controlled based on the following scenarios:

When *aql* is greater than *minthershold* and less than *maxthershold* and *ql* is greater than *minthershold*, ERED drops every arriving packet with probability D_P.

When *aql* is less than *minthershold* and the ql less than 1.75* maxthershold, D_p is increased.

As a result, the ERED method controls the congestion in the router buffer by controlling the packet dropping function both with *aql* and *ql* by stabilizing the *aql* between *minthershold* and *maxthershold*, but partially. The ERED's drawbacks are the parameterization and having many *thersholds*.

4. *Dynamic DGRED*

The proposed DGRED is an extension of GRED [31]. DGRED employs a dynamic *maxthreshold* and *doublemaxthershold* to control the congestion in the router buffer at the early stage before it overflows. The aim of the DGRED algorithm is to stabilize the aql using a new defined value called Target aql (Taql). Taql is calculated between the *minthershold* and *maxthershold* (Fig. 11). Another aim for the proposed DGRED is providing better performance results than other AQM algorithms.

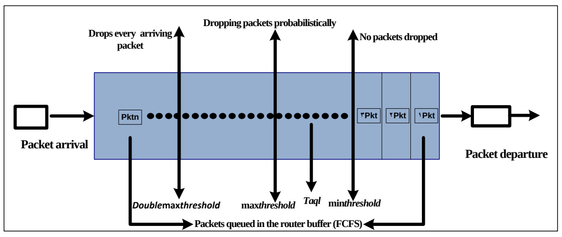

Figure 11. The Single Router Buffer For Proposed Dynamic GRED.

DGRED also updates the *maxthershold* and *doublemaxthershold* parameters at the router buffer to enhance network performance. DGRED uses the GRED algorithm's policy in dropping packets with probability when the aql is between the *minthreshold* and doublemaxthershold.

3. *EAGRED*

The Enhance AGRED [32], EAGRED is proposed is to improve the performance of AGRED when the congestion occur. In addition, The proposed algorithm aims to enhance the parameter settings, e.g stabilized the average queue length in the router buffer. Also, EAGRED control the congestion in the router buffer by controlling the dropping packets in the router buffer. See fig. 12.

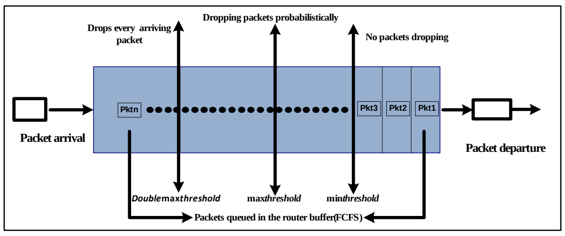

Figure 12. Single Router Buffer for EAGRED

The EAGRED algorithm scenario like AGRED algorithms, When *aql* at the router is below the *minthreshold*, no packets are dropped, When *aql* is between the min*th*reshold and *maxthreshold*, the router will drop the arriving packets randomly, similar to RED.

When *aql* is between the *maxthreshold* and the *doublemaxthreshold*, the packets are dropped randomly with a higher probability compared to the previous case. When *aql* is equal or greater than the *doublemaxthreshold*, the GRED router drops the arriving packets with D_p equal to one (i.e., arriving packets are dropped).

Unfortunately, EAGRED has some limitations. EAGRED deals with several threshold values and must set its parameters to specific values to obtain satisfactory performance (i.e., parameterization). Second, when the *aql* is less than the *minthreshold* and heavy congestion occurs, the *aql* will take time to adjust, during which the router buffer will likely overflow.

I. CONCLUSION AND FUTURE WORK

In this paper , we presented a survey on current advances in the area of active queue management . Parameter-based early congestion control methods are varied. Mostly, each of them built to solve some problems, but results in new limitations. Decbit method is very simple, distributed and has low computational overhead. However, this method takes time to adjust and react to congestion. RED, on the other hand, is the most stable method and has been adopted by the Internet Engineering Task Force (IETF) in RFC 2309 [6]. RED has used different reaction procedures to eliminate the slow congestion reaction. However, RED is still, to some extent, slow in adjusting and reacting to congestion and could not stabilize *aql* at a certain level when heavy congestion occurs. Moreover, RED is sensitive to parameters initialization. ARED eliminates some of the parameters sensitivity found in RED but could not stabilize *aql*. Similarly, SRED also eliminates some of the parameters sensitivity using the zombie list but it has a high packet loss rate. GRED stabilizes the *aql* at a certain level using three thresholds, but this has worsened the parameter sensitivity and could not stabilize *aql* when it is less than the *minthreshold* and heavy congestion occurs. DRED stabilizes the *aql* at defined level (T*ql*) regardless of the number of connections in the network, but it is sensitive to parameter initialization and adds unnecessary probabilistic packets dropping. BLUE avoids the bursty traffic flows by allowing space to accommodate the bursty packets in its router buffer. Yet, similar to DRED, it is sensitive to parameter initialization and adds unnecessary probabilistic packet dropping. REM stabilizes the (*ql*) at a certain level (target queue length (T_{ql})). It is also sensitive to parameter initialization, complex and has low throughput when the traffic is high. Adaptive Maximum Threshold stabilizes the *aql* value at T_{aql}, but it also suffers from parameter sensitivity and takes time to adjust. ERED controls the congestion in the router buffer by controlling the packet dropping function both with *aql* and *ql*. However, ERED is very sensitive to parameter initialization as it depends on many *thersholds*.

REFERENCES

[1] M. Welzl, *Network Congestion Control: Managing Internet Traffic*, 1 ed., 2005.

[2] B. Abbasov and S. Korukoglu, "Effective RED: An algorithm to improve RED's performance by reducing packet loss rate," *Journal of Network and Computer Applications*, vol. 32, pp. 703-709, 2009.

[3] M. E. Woodward, *Communication and Computer Networks: Modelling with discrete-time queues*: Wiley-IEEE Computer Society Press, 1993.

[4] G. F. A. Ahammed and R. Banu, "Analyzing the Performance of Active Queue Management Algorithms," *International Journal of Computer Networks & Communications* vol. 2, 2010.

[5] C. Kandaswamy and P. Ganapathi, "FloadAutoRED: an AQM scheme to Increase the Overall Performance in Internet Routers," *International Journal of Computer Science*, vol. 8, pp. 308-312, 2011.

[6] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, *Recommendations on Queue Management and Congestion Avoidance in the Internet*: RFC Editor,1998.

[7] C. Brandauer, G. Iannaccone, C. Diot, and S. Fdida, "Comparison of Tail Drop and Active Queue Management Performance for Bulk-Data and Web-Like Internet Traffic," in *Proceedings of the Sixth IEEE Symposium on Computers and Communications*: IEEE Computer Society, 2001.

[8] R. Stanojevic, R. N. Shorten, and C. M. Kellett, "Adaptive tuning of drop-tail buffers for reducing queueing delays," *Communications Letters, IEEE*, vol. 10, pp. 570-572, 2006.

[9] A. Bitorika, M. Robin, M. Huggard, and C. M. Goldrick, "A Comparative Study of Active Queue Management Schemes," in *Proceddings of IEEE ICC 2004, Congestion Control Under Dynamic Weather Condition 103*, 2004.

[10] J. H. Salim and U. Ahmed, *Performance Evaluation of Explicit Congestion Notification (ECN) in IP Networks*: RFC Editor, 2000.

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

[12] S. Floyd, R. Gummadi, and S. Shenker, "Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management," *AT&T Center for Internet Research at ICSI,* 2001.

[13] S. Athuraliya, S. H. Low, V. H. Li, and Y. Qinghe, "REM: active queue management," *Netwrk. Mag. of Global Internetwkg.*, vol. 15, pp. 48-53, 2001.

[14] D. Lapsley and S. Low, "Random early marking: an optimisation approach to Internet congestion control," in *Networks, 1999. (ICON '99) Proceedings. IEEE International Conference on*, 1999, pp. 67-74.

[15] W.-c. Feng, D. D. Kandlur, D. Saha, and K. G. Shin, "BLUE: A New Class of Active Queue Management Algorithms," University of Michigan, Ann Arbor, MI, Technical Report 1999.

[16] W.-c. Feng, S. K. G., K. D. D., and S. D., "The BLUE active queue management algorithms," *Networking, IEEE/ACM Transactions on,* vol. 10, pp. 513-528, 2002.

[17] W.-c. Feng, K. D. D., S. D., and S. K. G., "Stochastic fair blue: a queue management algorithm for enforcing fairness," in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 2001, pp. 1520-1529 vol.3.

[18] S. Floyd, "Recommendations On Using the Gentle Variant of RED," in *http://www.aciri.org/floyd/red/gentle.html*, 2000.

[19] J. Aweya, M. Ouellette, and D. Y. Montuno, "A control theoretic approach to active queue management," *Comput. Netw.*, vol. 36, pp. 203-235, 2001.

[20] T. J. Ott, T. V. Lakshman, and L. Wong, "SRED: stabilized RED," in *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 1999, pp. 1346-1355 vol.3.

[21] J. Ababneh, H., Thabtah, W., Hadi, E., Badarneh, , "Derivation of Three Queue Nodes Discrete-Time Analytical Model Based on DRED Algorithm". The Seventh IEEE International Conference on Information Technology: New Generations (ITNG 2010). IEEE Computer Society, pp. 885-890, April 2010, Las Vegas, USA.2010.

[22] H., Al-Bahadili, J., Ababneh, and F., Thabtah,"Analytical Modeling of a Multi-Queue Nodes Network Router, " International Journal of Automation and Computing (IJAC),Vol. 8, No.4, Springer, UK, 20/11/2011, pp.459 - 464.,2011.

[23] M. H. Yaghmaee and H. AminToosi, "A Fuzzy Based Active Queue Management Algorithm," *Computer Department, Ferdowsi University of Mashhad, Faculty of Engineering, Mashad,* pp. 458-462, 2003.

[24] C. Chrysostomou, A. Pitsillides, G. Hadjipollas, A. Sekercioglu, and M. Polycarpou, "Fuzzy Explicit Marking for Congestion Control in Differentiated Services Networks," in *Proceedings of the Eighth IEEE International Symposium on Computers and Communications*: IEEE Computer Society, 2003.

[25] K. K. Ramakrishnan and J. Raj, "A binary feedback scheme for congestion avoidance in computer networks with a connectionless network layer," in *Symposium proceedings on Communications architectures and protocols* Stanford, California, USA: ACM, 1988.

[26] D. Lin and R. Morris, "Dynamics of random early detection," *SIGCOMM Comput. Commun. Rev.*, vol. 27, pp. 127-137, 1997.

[27] W.-c. Feng, K. D. D., S. D., and S. K. G., "A self-configuring RED gateway," in *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 1999, pp. 1320-1328 vol.3.

[28] R. Morris, "Scalable TCP Congestion Control," in *Proceedings of the IEEE INFOCOM 2000 Conference*, 2000.

[29] K. Ramakrishnan, S. Floyd, and D. Black, *The Addition of Explicit Congestion Notification (ECN) to IP*: RFC Editor, 2001.

[30] A. Geat, M. E. Woodward, and M. Etbega, "Two Different Approaches of Active Queue Management," in *Networking, Sensing and Control, 2007 IEEE International Conference on*, 2007, pp. 579-583.

[31] M. Baklizi, H. Abdel-jaber, M. M. Abu-Alhaj, N. Abdullah, S. Ramadass, and A. ALmomani1, "Dynamic Stochastic Early Discovery: A New Congestion Control Technique to Improve Networks Performance," *International Journal of Innovative Computing, Information and Control* vol. 9, pp. 1-10, 2013.

[32] R. Kumar and J. Kesarwani, "EAGRED: A Enhance Version of Active Queue Managment Algorithms of Congestion Avoidence," *international Journal of Scientific Research And Education* vol. 2, 2014.