*International Journal of*
*Emerging Research in Management &Technology*
*ISSN: 2278-9359 (Volume-4, Issue-5)*

**Research Article** | **May 2015**

# Comparing Different Active Queue Management Techniques

**Nancy**
Research Scholar,
Department of Computer Science and Engineering,
Yamuna Inst. of Engg. and Tech.,
Gadholi, India

**Gurpreet Singh**
Dean Academics, Head (CSE), Associate Professor
Department of Computer Science and Engineering
Yamuna Inst. of Engg. and Tech.,
Gadholi, India

***Abstract-***

***Congestion is one of the biggest issues with Transmission Control Protocol (TCP) network environment. So various congestion control algorithms are studiedto keep the stability of the network. Therefore a router based mechanism called Active Queue Management (AQM) has been proposed to detect congestion early and to convey the congestion notification to sources before queue overflow and packet loss occurs. This is used by the routers to control the congestion where packets are dropped before the queues become full. A number of Active Queue Management algorithms such as random early detection (RED), Fair RED (FRED), Blue and Stochastic Fair Blue (SFB)have been studied. This paper presents a comparative study of these algorithms.***

***Keywords:Congestion Control, Networks, Queue Management, RED, Throughput, FRED, Blue, SFB, ECN***

## I. INTRODUCTION

Congestion in a network or internet creates problems such as reduced availability and throughput for the end user.When there are so many incoming packets contending for the limited shared resources, such as the queue buffer in the router and the outgoing bandwidth, congestion may happen in the data communication. During congestion, large amounts of packet experience delay or even be dropped due to the queue overflow[8].This will result in increasing packet loss rate and degradation of throughput. Congestion will also decrease efficiency and reliability of the whole network [8]. At very high traffic, performance collapses completely and almost no packets are delivered.

When a packet is dropped before it reaches its destination, all of the resources it has consumed in transit are wasted. As a result, many congestion control methods are proposed to solve this problem and avoid the damage. Most of the congestion control algorithms[3] are based on evaluating the network feedbacks to detect when and where congestion occurs, and take actions to adjust the output source, such as reduce the congestion window (cwnd)[7]. The feedbacks are used in the congestion detection and analysis. There are two categories of congestion control using feedback: Explicit feedback and Implicit Feedback.

In explicit feedback algorithms, some signal packets are sent back from the congestion point to warn the source to slow down[17] so as to reduce congestion while the implicit feedback algorithms, the source deduces the congestion existence by observing the change of some network factors, such as delay, throughput difference andpacketloss[17]. The Internet has mainly relied on thecooperative nature of TCP congestion control in order tolimit packet loss and fairly share networkresources.However, new applications are being deployed which donot use TCP congestion control and are not responsive tothe congestion control and are not responsive to thecongestion signals given by the network. Suchapplications are potentially dangerous because they driveup the packet loss rates in the network and can eventuallycause congestion collapse.

So Researchers andtheIETFproposeda mechanism: active queue management (AQM) for detecting congestion inside the network. Further, they have strongly recommended the deployment of AQMin routers as a measure to preserve and improve performance.Since AQM is a router based mechanism, so AQM algorithms [6] run on routers and detects congestion by typically monitoring the average queue size. When the average queue size exceeds a certain threshold, but is still less than the capacity of the queue, AQM algorithms infer congestion on the link and notify the end systems to back off by dropping some of the packets arriving at a router. Alternately, instead of dropping a packet, AQM algorithms can also set a specific bit in the header of that packet and forward the packet toward the receiver after congestion has been inferred. Upon receiving that packet, the receiver in turns sets another bit in its next ACK.

When the sender receives this ACK, it reduces its transmission rate as if its packet were lost. The process of setting a specific bit in the packet header by AQM algorithms and forwarding the packet is also called marking. A packet that has this specific bit turned on is called a marked packet. End systems that experience the marked or dropped packets reduce their transmission rates to relieve congestion and prevent the queue from overflowing. This paper presents a basic comparison of Active Queue Management Algorithms called random early detection (RED)[3], Fair RED (FRED)[6], Blue [9] and Stochastic Fair Blue (SFB) [10].

## II. LITERATURE SURVEY

One of the biggest problems with TCP's congestion control algorithm [5] over drop-tail queues is that the sources reduce their transmission rates only after detecting packet loss due to queue overflow. Since a considerable amountof time may elapse between the packet drop at the routerand its detection at the source, a large number of numbers of packets may be dropped as the senders continue transmission at a rate the network cannot support.

 

RED [3] starts to probabilistically drop packets long before the buffer is full, providing an early congestion indication to flaws which can then gracefully back off before the buffer overflows. RED maintains two bufferthresholds. When the exponentially averaged buffer occupancy is smaller than the first threshold, no packet is dropped, and when the exponentially averaged buffer occupancy is larger than the second threshold, all packets are dropped. When the exponentially averaged buffer occupancy is between the two thresholds, the packet dropping probability increases linearly with buffer occupancy. It is based on queue length as an estimator of congestion and also requires a wide range of RED parameters to operate correctly under different congestion scenarios. Unfortunately, when a large number of TCP sources [7] are active, the aggregate traffic generated is extremely bursty. Bursty traffic often defeats the active queue management techniques used by RED since queue lengths grow and shrink rapidly. While ECN is necessary for eliminating packet loss in the Internet, we show that RED, even when used in conjunction with ECN, is ineffective in preventing packet loss.

Feng et. al.[12] proposes Adaptive RED, which adjusts the packet dropping probability based on the past history of the average queue size and also proposes that Adaptive RED leaves the choice of the target queue size to network operators who must make a policy tradeoff between utilization and delay. We find that this revised version of Adaptive RED, which can be implemented as a simple extension within RED routers, removes the sensitivity to parameters that affect RED's performance and can reliably achieve a specified target average queue length in a wide variety of traffic scenarios. Based on extensive simulations, we believe that Adaptive RED is sufficiently robust for deployment in routers.

Feng et. al.[10] also proposes a mechanism for active queue management that is based not on the queue size, but only on buffer overflow and link idle events. In Stochastic Fair Blue (SFB), a variant of Blue, flashis hashed into accounting bins, and each bin has its own packet-marking probability.SFB scalable detects and rate-limits non-responsive flows through the use of a marking probability derived fromthe Bluequeue management algorithm and a Bloomfilter. In this, non-responsive flows were rate-limitedto a fixed amount of bandwidth across the bottlenecklink. However, it is possible to rate-limit nonresponsiveflows to a fair share of the link's capacity.

Suteret. al.[13] proposes that Per-flow service disciplines are only effective when supported by an appropriate per-flow buffer management strategy, and in particular the shared buffer with soft-partitioning, drop-from-front and longest queue drop.

Santhiet. al.[9] concludes that Blue uses the packet loss and link utilization history of congested queue, instead of queue lengths to manage congestion. A SFB, a technique using Blue for scalable and accurately enforcing fairness amongst flows in a large aggregate and shows that Blue is better than RED to avoid global synchronization for maintaining a single marking probability. And also, the results show that, among the five algorithms, SFB and CSFQ (Core Stateless Fair Queuing) are more effective at stabilizing the queue size and controlling the packet loss rate among non-responsive flows while maintaining high link utilization. The performance of SFB and CSFQ are obviously better than that of RED,FRED and Blue.

Smith et. al.[14] concludes that AQM can improve application and network performance for Web or Web-like workloads. In particular, it appears likely that with AQM and ECN(Explicit Congestion Notification), provider links may be operated at near saturation levels without significant degradation in userperceived performance.

Ali et. al.[6] compares the AQM algorithms based on simulation settings using RED and Drop Tail as baseline and design experiments to simulate the queue management techniques, and analyze the throughput, and fairness and found that RED performed slightly better with higher throughput and higher fairness Index than Droptail.

## III. ACTIVE QUEUE MANAGEMENT ALGORITHMS

To counteract the network performance deterioration caused by the increased load, newTCP specifications [16] have been proposed (e.g. Tahoe, Reno, Vegas), but they still only provide end-to-end congestion control functionality, keeping a drop-tail policy for routing. The new challenge has moved to the queue management: instead of waiting until congestion [18]is present, i.e. for timeouts or triple duplicate-acks to decrease the transmission window. The aim of Active Queue Management (AQM)[18] is to prevent full queues by signalling congestion to the transmitter before it happens.

The AQM algorithm [5] controls the arrival rate of packets into the queue, by ECN marking or packet dropping to generate the congestion signal that controls the source rate. In the optimization problem analogy, determining the right feedback signal is akin to finding the right price that controls the demand of the sources to match to the target link capacity. It has been shown that different AQMs solve the underlying utility optimization problem however with differences in properties such as the steady state backlog in the link, speed of convergence, transient response, and stability regions.

We will now examine these differences and discuss the aims of AQMs[18] in general. The aims of flow control include fairness, utilization, quality of service, good transient properties and scalability.

Fairness principally refers to the rate allocation that occurs at steady state, For the Internet, fairness is utility maximization. Utilization refers to the AQM's ability to drive the link at some target capacity, and controls application throughput. However, the greater the utilization, the greater the queuing delays experienced.

The design of the AQM algorithm has received particular attention of researchers, because it is currently the most significant bottleneck in the QoS performance of the best-effort Internet. Currently, the tail-drop queue is effectively the default and practically the only AQM on the Internet. Unfortunately, the tail-drop AQM [6] requires unnecessarily large backlogs and packet loss in queues and research points out that replacing this component alone would significantly improve the performance of the network. The various Active Queue management Algorithms are:

 

- RED (Random Early Drop)
- FRED (Fair RED)
- Blue
- SFB(Stochastic Fair Blue)

***(i) RED (Random Early Drop)***

The basic idea behind RED [4] queue management is to detect incipient congestion early and to convey Explicit congestion notification (ECN) [17] to the end-hosts, allowing them to reduce their transmission rates before queues in the network overflow and packets are dropped. The objectives for which RED was designed are to: (1) maintain high link utilization, (2) minimizes packet losses and queuing delay, (3) remove biases against bursty sources and (4) avoid global synchronization of sources.

To do these objectives, RED [1] maintains an EWMA(exponentially-weighted moving average) of the queue length to detect congestion. When the average queue length exceeds a minimum threshold (*Minth*), packets are randomly dropped or marked with an explicit congestion notification (ECN) bit. When the average queue length exceeds a maximum threshold (*Maxth*), all packets are dropped or marked.Since RED is an improvement over drop tail queues,it has few shortcomings also. One of the fundamental problems with RED is that it depends on queue length as an estimator of congestion. While the presence of a persistent queue indicates congestion, its length gives very little information as to the severity of congestion.

RED represents a class of queue management mechanisms that does not keep the state of each flow. That is, they put the data from the all the flows into one queue, and focus on their overall performance. Thus the problem of non-responsive flows originates.

**RED Algorithm:**

| Parameter | Meaning |
|---|---|
| Maxth | Maximum threshold |
| Minth | Minimum threshold |
| Maxp | Maximum packet Dropping/Marking Probability |
| Wq | Weighting factor |
| Pa | Probability |
| Avg | Average queue length |

/*RED Algorithm*/
*For each new arrival of packet:*
*Compute the average queue length;*
*If ( Minth ≤ Avg<Maxth)*
*{*
    *Calculate the probability Pa,*
    *with probability Pa: mark/drop the arriving packet*
*}*
*Else if (Maxth ≤ Avg)*
*{*
    *mark/drop the arriving packet*
*}*
*Else*
*{*
    *Do not mark/ drop the packet*
*}*

As the average queue size at the arrival of a new packet can be calculated by using the formula i.e.
*Avg = (1 – weight) X Avg + weight x currQ*
*Where 0 < Weight < 1*
*currQ is the current queue length*

***(ii) FRED (Flow Random Early Drop)***

Flow Random Early Drop (FRED) [6] is a modified version of RED, which eliminates the problem caused by non-responsive flows because it uses per-active flow accounting to make different dropping decisions for connections with different bandwidth usages. It only keeps track of flows that have packets in the buffer, thus the cost of FRED is proportional to the buffer size and independent of the total flow numbers.

Some interesting benefits of FRED include: (1) per-flow queuing and round-robin scheduling with substantially less complexity, (2) protecting fragile flows by deterministically accepting flows from low bandwidth connections, (3) penalizing non-adaptive flows by imposing a maximum number of buffered packets, and surpassing their share to

 

average per flow buffer usage,(4) providing fair sharing for large numbers of flows by using "two packet- buffer" when buffer is used up,(5) fixing several imperfections of RED[4] by calculate average queue length at both packet arrival and departure.
FRED uses two parameters: minq(minimum number of packets) and maxq(maximum number of packets that each flow is allowed to bufferFRED[6] also uses a global variable avgcq to track the average per active flow usage. It maintains the number of active flows, and for each of the flow, FRED maintains a count of buffer packets, *qleni*, and a count of times when the flow is not responsive (*qleni>maxq*). FRED penalize the flows with high strike values. FRED processes arriving packets using the following algorithm:
**/*** *FREDalgorithm* *******/*

***Global Variables:***
*avg: average queue size;*
*avgcq: average per-flow queue size;*
*maxq: maximum allowed per flow queue size;*

***Per-flow Variables:***
*qleni: number of packets buffered;*
*strikei: number of over-runs;*

```
Identify and manage non-adaptive flows:
If(qleni>=maxq || (avg>=maxth&&qleni>2*avgcq) || (qleni>=avgcq&&strikei>1))
{
        Strikei++;
        Drop packet p;
        Return;
}
If(minth<=avg<maxth)
{
        Operate in random drop mode
}
Elseif(avg<minth)
        No drop mode;
Else
{
        Drop-tail mode;
        Drop packet p;
}
```

***(iii) Blue***
Blue [9] performs queuing management based on packet loss and link utilization. Itmaintains a marking probability *pm* to either mark or drop the packets. If the queue iscontinually dropping the packets, *pm* is incremented by a factor d1. If the queue is emptyor link is idle, *pm* is decremented by a factor d2. The value of d1must be set significantlylarger than d2..This is because the link is underutilized when the congestion management iseither too aggressive or too conservative, but packet loss[6] occurs only when the congestionmechanism is too conservative. Blue uses one more parameter *freeze_time*, whichdetermines the time interval between two successive updates of *freeze_time*. It allows thechanges in the marking probability to take effect before the value is updated again. Bluealgorithm[19] is given below.

```
/*The Blue algorithm*/
Upon Packet loss (or Qlen>L) 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
```

Marking probability, *pm*, is also updated when the queue length exceeds a certain value, inorder to allow room to be left for transient bursts and to control the queueing delay whenthe size of the buffer being used is large.

***(iv) SFB***
Stochastic Fair Blue(SFB) [10] is a novel technique for enforcing fairnessamong a large number of flows. SFB scalable detectsand rate-limits non-responsive flows through the useof a marking probability derived fromthe Blue queuemanagement algorithm. It maintains few accounting bins. These bins are organized in *L* levels with *N* bins in each

 

level. Also, SFB maintains *L* independent hash functions, each associated with one level of the accounting bins [10]. Each hash function maps a flow into one of the accounting bins in that level. The accounting bins are used to keep track of queue occupancy statistics of packets belonging to a particular bin. As a packet arrives at the queue, it is hashed into one of the *N* bins in each of the *L* levels. If the number of packets mapped to a bin goes above a certain threshold (i.e., the size of the bin), the packet dropping probability *Pm* for that bin is increased.

If the number of packets in that bin drops to zero then*Pm* is decreased. It is observed that a non-responsive flow quickly drives *Pm* to 1 in all of the *L* bins it is hashed into. Responsive flows may share one or two bins with non-responsive flows, however, unless the number of non-responsive flows [6] is extremely large compared to the number of bins, a responsive flow is likely to be hashed into at least one bin that is not polluted with non-responsive flows and thus has a normal value. The decision to mark a packet is based on *Pmin*the minimum *Pm* value of all bins to which the flow is mapped into. If *Pmin*is 1, the packet is identified as belonging to a non-responsive flow and is then rate-limited.

The various parameters that are used in the SFB algorithm [10] are ql*en, Bin_Size, d1, d2, freeze_time, N, L, Hinterval*. *Boxtime*.*Bin_Size*is the buffer space for each bin. *Qlen*is the actual queue length of each bin. For each bin, *d1*, *d2* and *freeze_time*have the same meaning as that in Blue. Besides, *N* and *L* are related to the size of the accounting bins, for the bins are organized in *L* levels with *N* bins in each level. *Hinterval*is the time interval used to change hashing functions in our implementation for the double buffered moving hashing. *Boxtime*is used by penalty box of SFB as a time interval used to control how much bandwidth those non-responsive flows could take from bottleneck links.

```
/*SFB algorithm*/
Define B[l][n]: L x N array of bins (L levels, N bins per level)
At packet arrival:
enque()
        Calculate hash function values h0, h1, ..., hL-1;
        Updateqlen for each bin at each level
        fori = 0 to L-1
        if (B[i][hi].qlen>bin_size)
                B[i][hi].pm += delta;
                Drop packet;
        else if (B[i][hi].qlen == 0)
                B[i][hi.:pm -= delta;
        pmin = min(B[0][h0]:pm .. B[L][hL]:pm);
        if (pmin ==                     1)
                ratelimit()          // limit flow's sending rate
        else
                Mark/drop packet with probability pmin;
```

## IV. COMPARISON

It is hard to conclude which algorithm is better in all aspects than another, especially considering the deployment complexity.So the major trends which are observed are: (1) all these algorithms provide high link utilization, (2) RED and Blue don't identify and penalize non-responsive flow, while the other three algorithms maintains fair sharing among different traffic flows, (3) the fairness is achieved using different methods, FRED record per-active-flow information, SFB statistically multiplex buffers to bins, but needs to be reconfigured with large number of non-responsive flows (4) all of the algorithms has computation overhead per incoming packet, their space requirements are different. The following table summarizes our evaluation results:

Table 1: Comparison among RED, FRED, Blue and SFB

| **Properties** | **RED** | **FRED** | **Blue** | **SFB** |
|---|---|---|---|---|
| **Goal of Algorithm** | Optimized for cell-based architecture in ATM networks. | Make RED fair. | Low loss rates and low queue length oscillation. | To detect the non-responsive flows |
| **Fair bandwidth allocation** | Depends on strategy, ranging from unfair to fair. | Yes, $P_m$ is proportional to bandwidth utilization per flow. | Yes | Yes |
| **Malicious-aware** | No, $P_{drop}$ increases linearly with bandwidth utilization. | No | No | No |
| **Link Utilization** | Good | Good | Good | Good |

 

| | | | | |
|---|---|---|---|---|
| **Target quality interval** | Depends on strategy, from no target to dynamic target. | Static target interval [Qmin avg ,Qmax avg ] for Qavg. | No, just minimize loss rate and queue size. | --- |
| **Space Requirement** | Large | Small | Small | Large |
| **Configuration Complexity** | Hard | Easy | Easy | Hard |
| **Special characteristic** | Takes the characteristics of ATM networks into account. | RED combined with per-flow state. | Hash-bin based detection of greedy flows | Enforce fairness among a large number of flows. |

## V. CONCLUSION

From the study, we can clearly see that the RED algorithm does not perform well as the Blue control algorithms in the heavily loaded network. In addition, RED is difficult to stabilize the queue size. The parameter configuration of RED is another challenge task under different network scenarios and over a wide range of load levels. Blue uses the packet loss and link utilization history of congested queue, instead of queue lengths to manage congestion. Blue also has difficult to stabilize the queue size. It needs some starting time for Blue to reach a fine-tuning. To solve this problem, another researcher proposed another enhancedBlue algorithm, Stochastic Blue queuing algorithm. SFB is also called fair Bluequeuing algorithm. SFB is a technique using Blue for scalable and accurately enforcing fairness amongst flows in a large aggregate. Using SFB, non-responsive flows can be identified and rate-limited using a very small amount of state. SFB does not rely on coordination between intermediate routers and edge markers and can perform will without placing additional overhead in packet headers.

## REFERENCES

[1] S. Floyd, R. Gummdi, and S. Shenker, "Adaptive RED,Analgorithm for increasing the robustness of RED", AT&TCenterforInternetResearchatICSI,Aug. 2001.

[2] Mahmud Etbega Mohamed, M.E. Woodward, "Adaptive Early Random Drop: An algorithm for controlling queueing delay in a buffer", Proceeding of 16th Telecommunication Forum IEEE, Belgrade-Serbia, 2008.

[3] S.Floyd and V. Jacobson, "Random early detection for congestion avoidance", IEEE/ACM Transactions Networking, Vol. 1,issue 4, pp 397-413, July 1993.

[4] T. Bonald, M. May, and J. Bolot. "Analytic evaluation of RED performance.", in INFOCOM, vol.3,pp 1415–1424, 2000.

[5] W. Feng, "Improving Internet Congestion Control and Queue Management Algorithms", PhD Thesis, University of Michigan, 1999.

[6] Dr. T. BhaskarReddy, Ali Ahammed, and Reshmabanu, "Performance Comparison of Active Queue Management Techniques",IJCSNS vol.9, Issue 2, pp405-408,February 2009.

[7] Amanpreet Kaur, Gurpreet Singh, Baninder Singh, "Evaluation of Congestion control variants of TCP by strolling propagation delay in NS-2", International Journal for Applied Engineering and Research (Vol. V) with print ISSN No: 0973-4562 and Online ISSN No: 1087—1090, April, 2011.

[8] Santhi,V., Natarajan,A.M,"Performance Analysis of Active Queue Management Algorithms",International Journal on Information Sciences and Computing, Vol.3, Issue 1, January 2009.

[9] W. Feng, D. Kandlur, D. Saha, and K. G. Shin, "The Blue active queue management", IEEE/ACM transactions on networking, vol 10, Issue 4, August 2002.

[10] W. Feng, D. Kandlur, D. Saha, and K. Shin, "Stochastic fair Blue: A queue management algorithm for enforcing fairness", in INFOCOM, pp 1520–1529, Los Alamitos, CA, Apr 2001.

[11] Ion Stoica, S. Shenker, and H. Zhang, "Core-Stateless Fair Queueing: A Scalable Architecture to Approximate Fair Bandwidth Allocations in High Speed Networks", Journal of IEEE/ACM Transactions on Networking (TON), Volume 11, Issue 1,pages 33-46, February 2003.

[12] W. Feng, D. Kandlur, D. Saha, and K. G. Shin, "Techniques for eliminating packet loss in congested TCP/IP networks", Univ. Michigan, Ann Arbor, MI, Tech. Rep.UMCSE-TR-349-97, Oct. 1997.

[13] Bernhard Suter, T. V. Lakshman, Dimitrios Stilidis, and Abhijit Choudhury, "Efficient Active Queue Management for Internet Routers", November 1997, Interop '98

[14] L. Le, J. Aikat, K. Jeffay, and F.D. Smith, "The Effects of Active Queue Management and Explicit Congestion Notification on Web Performance", IEEE/ACM Transactions on Networking, Volume 15 Issue 6,Pages 1217-1230, December 2007.

 

[15] Bartek Peter Wydrowski , "Techniques in Internet Congestion Control", Ph.D. Thesis, Electrical and Electronics Engineering Department, The University of Melbourne ,Febuary 2003.

[16] Amanpreet Kaur, Gurpreet Singh, Deepti Chauhan, "Radical Analysis of TCP Alternatives: Tahoe, New Reno, Sack, Vegas on the basis of imitation in NS-2", in the proceedings of Journal of Yamuna Journal of Technology and Management, Vol 1, Issue 1, Nov, 2011.

[17] L. Brakmo and S. O'Malley, "TCP-Vegas: New techniques for congestion detection and avoidance", In ACM SIGCOMM'94, pp. 24-35, OCT, 1994.

[18] Kamalpreet Kaur,Navdeep Kaur,Gurjeevan Singh, "Performance comparison of queuing algorithms: a review", IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)e-ISSN: 2278-2834,p-ISSN: 2278-8735.Volume 8, Issue 4 PP 46-48,Nov. -Dec. 2013.

[19] Vandana Khare,Dr. Y. Madhavee Latha, Dr. SrinivasRao, "Comparative Analysis of Queuing Mechanism for WCDMA Networks",International Journal of Emerging Trends & Technology in Computer Science (IJETTCS),Volume 2, Issue 2, March –April 2013.

 