RESEARCH ARTICLE OPEN ACCESS

Evaluation of Active Queue Management Algorithms

Arsh Arora[1], Lekha Bhambhu[2]
Research Scholar[1], HOD[2]
Department of Computer Science and Engineering
Jan Nayak Ch. Devi Lal Memorial Enginerring College, Sirsa
Haryana-India

ABSTRACT
As the internet is evolved, the users have dramatically increases, as the demand increases than available resources, congestion increases. The congestion creates many problems like data loss, long delay, wastage of resources and many more. In congestion, Active Queue Management (AQM) algorithms are very utilizing schemes. In order to reduce the increasing packet loss rates caused by an exponential increase in network traffic, researchers have been considering the exploitation of active queue management algorithms. In this paper we will discuss about congestion, congestion management and evaluate the active queue management algorithms such as Droptail, RED, RRED, WRED, ARED BLUE, REM,. These algorithms have been selected amongst the many published over the past few years, will be described in a simplified manner.
Keywords:- AQM, Congestion, Congestion Management, TCP, Droptail, RED, RRED, WRED, ARED, BLUE, REM

I. INTRODUCTION

The Internet and wireless technologies are developing rapidly and have been a magnificent success in the past few years. Its presence in everyday life is a fact. Traditional slow speed networks have been enforced to merge with the high speed networks. But due to increase in Internet size and no. of users, users are likely to experience longer delay, more packet loss and other performance humiliation issues because of network congestion. It has a huge influence to both wired network and wireless network and causes the problem of packet loss, packet delay and lock out. To control congestion we have to deploy congestion management. Congestion management features allow us to control congestion by determining the order in which packets are sent out an interface based on priorities assigned to those packets. To control congestion there are many techniques, such as exponential back off, congestion control in TCP, priority schemes and queue management techniques. To reduce the increasing packet loss rates caused by an exponential increase in network traffic, researchers have been taking into consideration the implementation of active queue management algorithms (AQM).

AQM is a router-based congestion control technique wherein routers notify end-systems of emerging congestion. All AQM designs function by detecting impending queue build up and notifying sources before the queue in a router overflows. The various designs proposed for AQM differ in the mechanisms used to detect congestion and in the type of control mechanisms used to achieve a stable operating point for the queue size. The basic goal of all AQM techniques is to keep the average queue size in routers small [5]. This has a number of desired effects including (1) Controls average queue size, (2) Absorbs bursts without dropping packets, (3) Prevents bias against bursty connections, (4) Avoids global synchronization of TCP, (5) Reduces the number of timeouts in TCP, and (6) Take actions against misbehaving flows.

II. CONGESTION MANAGEMENT

Congestion management [2] features allow you to control congestion by determining the order in which packets are sent out an interface. Congestion management necessitates the formation of queues, allocation of packets to those queues based on the specification and scheduling of the packets in a queue for transmission. During periods with light traffic, when there is no congestion departs, packets are sent out the interface as soon as they reach. During periods of transmit congestion at the outgoing interface, packets reach faster than the interface can send them to their destination. By using congestion management features, packets build up at an interface are queued until the interface is free to send them; they are then scheduled for transmission according to their assigned priority or the queuing mechanism configured for the interface. The router regulates the of packet transmission by controlling which packets are placed in which queue and how queues are serviced with respect to each other.

For congestion management there are many techniques, such as exponential back off, congestion control in TCP, priority schemes and queue management techniques:

.

A. Exponential Back off

Exponential back off is used in CSMA for Congestion Avoidance, which is sensing schema of 802.11. The sender senses the channel before the transmission of data. If the channel is busy it wait until idle and sends the data after a random period of time. The random period is calculated by exponential back off [1].

Congestion control in TCP consists of slow start, fast transmission, fast recovery, congestion avoidance [3]. It is a method to controlling the transmission rate of the sender. The TCP flows starts at a very slow rate and increase exponentially to a threshold. Congestion avoidance then happens and congestion window increases by one segment each time for one successful transmission.

B. *Congestion Control in TCP*

There are four congestion control algorithms are now in common use. Each of the algorithms described in this paper was actually established long before the standard was published [3]. The four algorithms described below are Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery.

1) *Slow Start:*

Slow Start, is TCP software implementations mechanism used by the sender to control the transmission rate, and then rate of acknowledgements returned by the receiver determine the rate at which the sender can transmit data. When a TCP connection starts, the Slow Start algorithm establishes a congestion window to a segment, which is the maximum segment size (MSS) initialized by the receiver during the connection establishment phase. The congestion window increases by one segment for each acknowledgement send by the receiver to sender. Thus, the sender can transmit the minimum of the congestion window and the advertised window of the receiver is known as transmission window [4].
Slow Start is actually is worthy when network is not congested and response time is good. When the congestion window may become too large for the network or network conditions may change, in that scenario packets may be dropped. Packets lost will trigger a timeout acknowledgement at the sender. After that, the sender goes into congestion avoidance mode as described in the next section.

2) *Congestion Avoidance:*

Congestion Avoidance is used to slow the transmission rate of packets. However, Slow Start is used in aggregation with Congestion Avoidance as the means to get the data transfer going again so it doesn't slow its transmission rate and stay slow. In Congestion Avoidance algorithm a retransmission timeout or the receiving of duplicate ACKs can indirectly points to the sender, that network congestion is taking place. The sender immediately decreases its transmission window size to half of its current size (the minimum of the congestion window size and the receiver's advertised window size). If congestion was occurred by timeout, the congestion window is reset to one segment, which puts the sender into Slow Start mode automatically. If congestion was occurred by duplicate ACKs, then Fast Retransmit and Fast Recovery algorithms are invoked (we will discuss them in next section). As the data is received by destination during Congestion Avoidance in network, the congestion window is increased. However, Slow Start is only used till the halfway point where congestion originally started [4].
After this halfway point, congestion window is increased by one segment for all acknowledged segments in transmission window. In this mechanism sender will be forced to slow its transmission rate, so that it will approach the point where congestion had been detected.

3) *Fast Retransmission:*

The TCP receiver sends duplicate ACK whenever the out of order segment reaches. The duplicate ACK is used to indicate the sender that an out-of-order segment is received. From the sender's perspective duplicate ACK can be received by number of network problems. These ACKs can be caused by dropped segments, re-ordering of data segments by the network, data segments by the network or replication of ACK. A TCP receiver should send an immediate ACK when the incoming segment fills in all or part of a gap in the sequence space. This will generate information more timely to the sender, so that sender can recover a loss through a fast retransmit, a retransmission timeout, an experimental loss recovery algorithm, such as NewReno [FH98].
The fast retransmit algorithm uses the 3 duplicate ACKs as an indication that a segment has been lost or damaged. After receiving 3 duplicate ACKs, TCP performs a retransmission of the missing segment, without waiting for expiration of the retransmission timer.

4) *Fast Recover:*

Since, fast retransmit algorithm sends the missing segment, but the fast recovery algorithm governs the new data until a non-duplicate ACKs reaches; Fast recovery is an improvement that allows high throughput under moderate congestion, specifically for large windows [4].
To summarize this section of the paper, figure 1 below shows what a TCP data transfer phase with TCP congestion control might look like. Notice the periods of exponential window size increase, linear increase and drop-off. Each of these scenarios shows the sender's response to implicit or explicit signals it receives about network conditions.

C. *Priority Queues:*

Priority queue schemes allow defining how traffic is prioritized in the network. Configure the traffic priorities, the queue with the highest priority is serviced first until it is empty, then the lower queues are serviced in sequence [2]. During transmission, priority is given to the queues absolute preferential treatment over low priority queues; we can give the highest priority to the important traffic, which always takes precedence over less important traffic. Packets are classified based on user-specified criteria and placed into one of the four output queues- high, medium, normal, and low—based on the assigned priority. Packets that are not classified by priority fall into the normal queue.

D. *Queue Management Techniques:*

Priority queue schemes allow defining how traffic is prioritized in the network. Configure the traffic priorities, the queue with the highest priority is serviced first until it is empty, then the lower queues are serviced in sequence [2]. During transmission, priority is given to the queues absolute preferential treatment over low priority queues; we can give the highest priority to the important traffic, which always takes precedence over less important traffic. Packets are classified based on user-specified criteria and placed into one of the four output queues- high, medium, normal, and low—based on the assigned priority. Packets that are not classified by priority fall into the normal queue.

III. ACTIVE QUEUE MANAGEMENT

An advancement of the router based queue management is known as Active Queue management. Generally, AQM schemes controls the congestion by controlling flow. Congestion is measured and control actions are taken. There are two approaches for measuring congestion [6].

Queue based: In queue based AQMs congestion is measured by queue size and action is taken by maintaining a set of queues by Internet routers, one per interface, that hold packets scheduled to start extinct on that interface. In such queues a packet is set onto the queue if the queue is shorter than its upper limit size, and dropped otherwise. The limitation of this is that a backlog of packets is inherently required by the control mechanism when the congestion is observed in queue is already positive.

Flow based: In Flow based AQMs, congestion is observed and action is taken based on the packet arrival rate. For such schemes, backlog, and all its unfavorable implications, is not necessary for the control mechanism.

There are many AQM schemes that have proposed in the literature. Here are the recently proposed AQM algorithms.

- Drop tail
- Random early detection (RED)
- Robust Random Early Detection (RRED)
- Weighting Random Early Detection (WRED)
- Adaptive Random Early Detection (ARED)
- BLUE
- Random Exponential Marking (REM)

A. Drop Tail

Tail Drop is the default congestion avoidance mechanism. It also impacts on the efficiency of network bandwidth utilization. When the Output Queue is filled with the packets and some of them arrive in on the Input Queue, then the packets which are arriving on the interface will be dropped. It does not matter whether it is a voice packet or a data packet, all packets will be dropped by default when Tail Drop is in action. This method has served the Internet well for years, but it has three important drawbacks [5].

Lock-Out: In some situations drop tail allows a single connection or a few flows to control queue space, preventing other connections from getting room in the queue. This "lock-out" phenomenon is often the result of synchronization or other timing effects.

Full Queues: The drop tail discipline allows queues to maintain a full (or, almost full) status for long periods of time, since tail drop signals congestion (via a packet drop) only when the queue has become full. It is important to reduce the queue size, and this is perhaps queue management's most important goal.

Global TCP Synchronization: When TCP Slow Start strike out, all senders on the network back off and you can see a drop in the bandwidth, then slowly everyone starts sending packets at higher rate as they find out no more packet loss, so all senders on the network starts sending the packets again at higher rate and you see peaks in the network bandwidth. At this time the interfaces can get congested again and packets can be dropped, which then makes all senders to drop their sending rate and wait for certain time interval where they see no more packet loss, this leads TCP Senders to again increase the sending rate. This goes on in cycles and this behaviour means a lot of bandwidth is just getting wasted. If you are monitoring the bandwidth with a graph, you will something like below graph in the utilization charts.

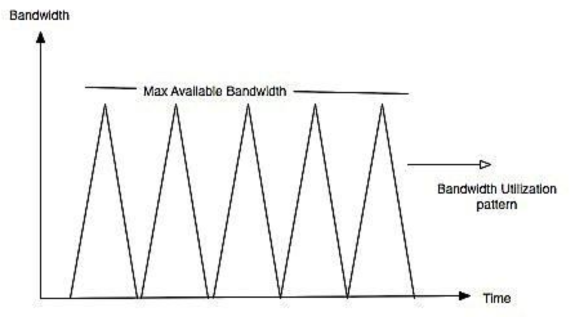

Fig. 1: Drop Tail mechanism of bandwidth utilization

This behaviour is also called as “Global TCP Synchronization” and it is responsible for a lot of network bandwidth wastage.

Nevertheless, drop tail has some weakness, such as the bad fairness, sharing among TCP connections and the throughput and link efficiency suffer severe degradation if congestion is making worse.

B. Random Early Detection (RED)

Random Early Detection (RED) was proposed by Floyd and Jacobson as an efficient congestion avoidance mechanism in the network routers/gateways. It also helps to prevent the global synchronization in the TCP connections sharing a congested router and to decrease the bias against bursty connections. It is assumed to solve the traditional problems of queue management techniques. It was an improvement over the previous techniques such as Random Drop and Early Random Drop [7]. RED use probabilistic discard methodology of queue fill before overflow conditions are reached. By detecting incipient congestion early and to convey congestion notification to the end-hosts, allowing them to decrease their transmission rates before queues in the network overflow and packets are dropped.

The RED gateway computes the average queue size by using a low pass filter along with an exponential weighted moving average. The average queue size is compared with two thresholds: a *minimum* and a *maximum* threshold. When the size of average queue is less than the minimum threshold, no packets are marked. When the size of average queue is greater than the maximum threshold, every arriving packet from gateway is marked. If marked packets are, in fact, dropped or if all source nodes are collaborative, this assures that the average queue size does not significantly exceed the maximum threshold.

When the average queue size is varying in between the minimum and maximum thresholds, each arriving packet is marked with a probability p_a, where p_a, is a function of the average queue size avg. Each time a packet is marked, the probability that a packet is marked from a particular link is roughly relative to that connection's share of the bandwidth at the gateway. The general RED algorithm is given below:

```
For each packet arrival
calculate the average queue size $avg$

if  $min_{th} \leq avg < max_{th}$

calculate probability $p_a$
      with probability $p_a$:
            mark the arriving packet
else if $max_{th} \geq avg$

      mark the arriving packet
```

Fig. 2. General algorithm for RED gateways [10]

Thus, the RED gateway has two separate algorithms. One of those computes the average queue size determines the degree of burstiness that will be allowed in the gateway queue. And the other one calculates the packet-marking probability that determines how often the gateway marks packets; give the current level of congestion. The goal of gateway is to mark the packets at fairly evenly spaced intervals, in order to avoid biases and avoid global synchronization, and to mark packets sufficiently frequently to control the average queue size.

C. Robust Random Early detection (RRED)

RED can detect and respond to long-term traffic patterns, but it cannot detect congestion caused by short-term traffic load changes. In addition, it is well known that an appropriate tuning of RED parameters is not an easy task and may result in a non-stabilizing controls scheme. Robust random early detection (RRED) [8] is a queuing discipline for a network scheduler. The existing random early detection (RED) algorithm and its variants are found vulnerable to emerging attacks, especially the Low-rate Denial-of-Service attacks (LDoS). Experiments have confirmed that the existing RED-like algorithms are notably vulnerable under LDoS attacks due to the oscillating TCP queue size caused by the attacks. The Robust RED (RRED) algorithm was proposed to increase the efficiency of TCP throughput against LDoS attacks. The basic idea behind the RRED is to detect and filter out attack packets before a normal RED algorithm is applied to incoming flows. RRED algorithm can significantly improve the performance of TCP under Low-rate denial-of-service attacks [9].

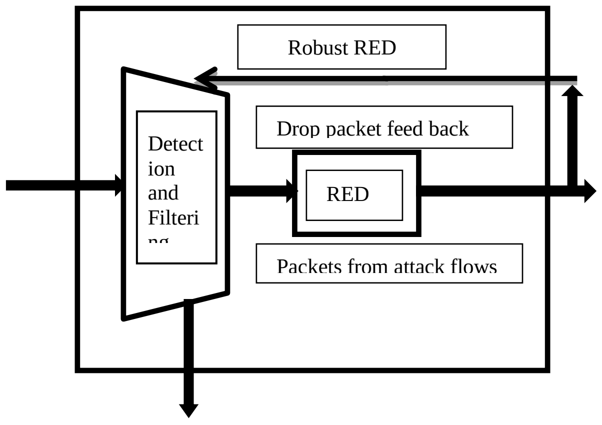

Fig.3 - Architecture of Robust RED

A detection and filter block is added in front of a regular RED block on a router. The basic idea behind the RRED is to detect and filter out LDoS attack packets from incoming explosions before they feed to the RED to a very hungry monkey algorithm. How to distinguish an attacking packet from normal TCP packets is critical in the RRED design.

Within a benign TCP flow, the sender will delay sending new packets if loss is detected (e.g., a packet is dropped). Consequently, a packet is suspected to be an attacking packet if it is sent within a short-range after a packet is dropped. This is the basic idea of the detection algorithm of Robust RED (RRED).

D. Weighted Random Early Detection (WRED)

By randomly dropping packets precedence to periods of high congestion, WRED tells the packet source to decrease the rate of its transmission. WRED drops packets built on IP precedence. Packets with a higher IP precedence are less expected to be dropped than packets with a lower precedence. WRED can selectively discard lower priority traffic when the interface begins to get congested and provide differentiated performance characteristics for different classes of service. By dropping some packets early rather than waiting until the queue is full, WRED avoids dropping large numbers of packets at once and minimizes the chances of global synchronization [10].

For interfaces configured to utilize the Resource Reservation Protocol (RSVP) attributes, WRED chooses packets from other flows to drop rather than the RSVP flows. Also, IP Precedence controls which packets are dropped—traffic that is at a lower precedence has a higher drop rate and therefore is more likely to be choked back.

WRED be at variance with other congestion avoidance techniques such as queuing strategies because it attempts to anticipate and avoid congestion rather than control congestion once it occurs.

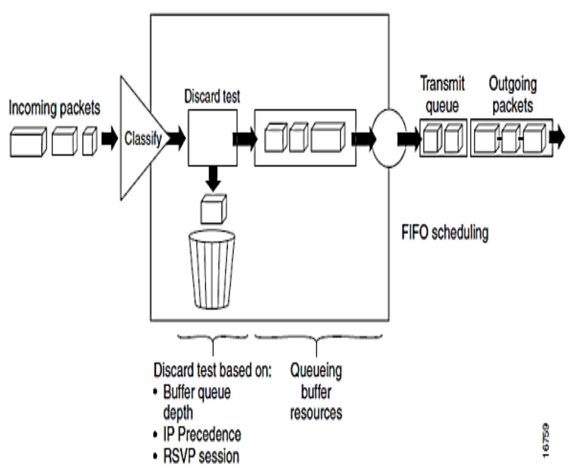

Fig4: Weighted Random Early Detection

WRED and distributed WRED (DWRED)—both of which are the Cisco implementations of RED. Within the section on WRED, the following related features are discussed:

- Flow-based WRED. Flow-based WRED extends WRED to provide greater fairness to all flows on an interface in regard to how packets are dropped.
- DiffServ Compliant WRED. DiffServ Compliant WRED extends WRED to support Differentiated Services (DiffServ) and Assured Forwarding (AF) Per Hop Behaviour (PHB). This feature enables customers to implement AF PHB by coloring packets according to differentiated services code point (DSCP) values and then assigning preferential drop probabilities to those packets.

E. Adaptive Random Early Detection (ARED)

ARED algorithm accomplishes whether to build RED more or less violent based on the observation of the average queue length [11]. If the average queue length moves back and forth around max_{th} then early detection is too violent. On the other hand if the average queue length moves back and forth around max_{th} then early detection is being too traditionalist. The algorithm changes the probability according to how violent it senses it has been removing traffic. So, adapting the RED parameter max_p and automatically setting the RED parameters w_q and max_{th} maintains a predictable average queue size and reduces RED’s parameter sensitivity. Adaptive RED, however, leaves the choice of the target queue size to network operators who must make a policy trade-off between utilization and delay.

F. BLUE

The blue algorithm resolves the shortcomings of RED algorithm by employing the hybrid control scheme with queue size congestion measuring scheme. It uses flow and queue events to modify the congestion notification rate. This rate is regulated by packet loss from queue congestion and link utilization. The key difference between Blue from red is that uses packet loss rather than average queue length [13].

BLUE maintains a single probability, P_m, to mark or drop packets. If the queue frequently dropping packets due to buffer overflow, BLUE increases P_m, thus increasing the rate at which it sends back congestion notification or dropping packets. On the other hand, if the queue is empty or if the link becomes idle, BLUE decreases its marking probability P_m. This effectively allows BLUE to “learn” the correct rate it needs to send back congestion notification or dropping packets (Feng, 2002/b).

BLUE typically depends upon two parameters that are d_1, d_2 and $freeze\_time$. $d_1$determines the amount by which P_m is increased when the queue overflows, while d_2 determines the amount by which P_m is decreased when the link is idle. $freeze\_time$ determines the minimum time interval between two successive updates of P_m. This allows the changes in the marking probability to take effect before the value is updated again. Based on those parameters the basic blue algorithms can be summarized as :

Upon packet loss event:	**Upon link idle event:**
If((now-$last\_update$) > $freeze\_time$) $P_m = P_m + d_1$; $last\_update$ = now;	if ((now - $last\_update$) > $freeze\_time$) $P_m = P_m - d_2$; $last\_update$ = now;

The BLUE algorithm

G. Random Exponential Marking (REM)

REM [1] is both a set of AQMs and a different technique for communicating congestion information. REM embodies a mechanism for the accurate communication of link congestion prices, so that the link congestion state covariant is exactly the congestion price as in the utility maximisation. A REM link indicate a packet at link l with a possibility based on the link price p_1 state, and a global encoding constant $\emptyset(1 < \emptyset)$

$$M_1(t) = 1 - \emptyset^{-pt}$$

Because sources know the usefulness of $\emptyset$, they can calculate

the total end-to-end path congestion price. Therefore, in a absolute deployment, REM requires a REM link algorithm and a source algorithm able of decoding REM information. It has been shown that inter-operation with the TCP-RENO source algorithm with just the link REM AQM algorithm [5] deployed is possible. In this case, the price $p_1(t)$ state covariant can be interpreted as the marking rate, just as the former AQMs discussed. I definitely for $p_1(t)$ << 1, we can assume $p_1(t) = m_1(t)$ by (1). With this in mind, the three control laws PC1, PC2, PC3 can be interpreted as alternative AQMs:

PC1 : $p_1(t+1) = \gamma b_1(t)$ (2)

PC2 : $p_1(t+1) = [p_1(t) - \gamma(x^1(t) - c_1]^+$ (3)

PC3: $p_1(t+1) = [p_1(t) - \gamma(\alpha_1 b_1(t) + x^1(t) - c_1]^+$ (4)

Where $p_1(t)$ is the congestion notification rate, c_1 is a target capacity just under the actual link capacity, $b_1(t)$ is the backlog, and α and γ are control gain constants which influence speed and firmness of control. It is clear that PC1 control law is parallel to the RED-like AQMs where the congestion notification rate is proportional to backlog. The control laws PC2 and PC3 that give a new approach to AQM design. PC2 and PC3 disengage the congestion notification rate from the backlog at the link. PC2 and PC3 evaluate the arrival rate to the link to compute the congestion notification rate instead of using the backlog. The congestion notification rate is measured by an integral controller, whose error term is the inconsistency between the cumulative arrival rate to the link and the target link capacity. Note that PC2 and PC3 differ only in that PC3 adds a backlog penalty term to the control process; if there is a backlog then it makes the marking rate increase with greater rate. This was set up to improve the transient response of the PC2 controller, and decrease the amount of backlog during transient periods when the load alters. The stability properties of PC3 are evaluated in. Further work in this area expands the analysis and improves the framework of REM. Several of author's papers focus on improving the convergence rate of the basic REM algorithm. With a faster rate of convergence of arrival rate to the target rate, jitter and the buffer requirements are reduced. An enhancement to the control equation founded on a Newton-like algorithm is evaluated. An approach using a deadbeat controller is used. Experimental results shows that the control laws PC2-PC3 are able to control the sources such so that the mean backlog at the link is significantly reduced compared to a tail-drop queue or RED. The result shows that these AQMs are able to works with very low backlog, and preserve a high link utilisation. The PC2-PC3 advance significantly both in design and performance from the RED or tail-drop algorithm.

IV. CONCLUSIONS

In this paper, we have mentioned the terms Congestion management and AQM (Active Queue Management). We have explained the main goals of AQM. In this work, the performance of seven AQM schemes, selected from amongst the many published over the past ten years has been evaluated. We have compared Droptail, RED, RRED, WRED, ARED, BLUE and REM algorithms AQM algorithms are absolutely useful because the management of packets to avoid congestion occasionally requires exceeding hardware capabilities. As long as this demand exceeding of hardware capabilities continue AQM algorithms will be popular and studies on networking flows area will go on.

ACKNOWLEDGMENT

Special thank you goes to those who contributed to this paper: Mrs. Lekha Bhambhu for his valuable comments and sharing her knowledge. Mr. Deepak Mehta & Mr. Gopal Sharma for their keen interest, expert guidance, valuable suggestions and persistent encouragement.

REFERENCES

[1] Long L., Aikat J., Jeffay K. and Smith F. 2005.The "Effects of Active Queue Management and Explicit Congestion Notification on Web Performance". IEEE/ACM Transactions on Networking.

[2] "Congestion management Overview", Cisco IOS Quality of Service Solutions Configuration Guide.

[3] Jacobson v. , August 1988 "Congestion Avoidance and Control", Computer Communications Review, Volume 18 number 4, pp. 314-329.

[4] M. Allman, et. al, "TCP Congestion Control", RFC 2581, 2001.

[5] B. Braden, et al, Recommendations on Queue Management and Congestion Avoidance in the Internet, RFC 2309, April, 1998.

[6] Serhat ÖZEKES, "EVALUATION OF ACTIVE QUEUE MANAGEMENT ALGORITHMS", stanbul Ticaret Üniversitesi Fen Bilimleri Dergisi Yıl:4 Sayı:7 Bahar 2005/1 s.123-140

[7] Floyd, S., and Jacobson, V. (1993), Random Early Detection gateways for Congestion Avoidance V.1 N.4, August 1993, pp. 397-413..

[8] Changwang Z., Jianping Y., Zhiping C., and Weifeng C (2010), RRED: Robust RED Algorithm to Counter Low-Rate Denial-of-Service Attacks IEEE COMMUNICATIONS LETTERS, VOL. 14, NO. 5, MAY 2010

[9] H. V. Shashidhara, Dr. S. Balaji, "Low Rate Denial of Service (LDoS) attack – A Survey", IJETAE, Volume 4, Issue 6, June 2014

[10] Congestion Avoidance Overview", Cisco IOS Quality of Service Solutions Configuration Guide QC-175, Weighted Random Early Detection

[11] Sally Floyd, Ramakrishna Gummadi, and Scott Shenker, "Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management", AT&T Center for Internet Research at ICSI August 1, 2001, under submission.

[12] Feng W., Shin K. G., Kandlur D. D., Saha D., (2002/b), "The BLUE active queue management algorithms", IEEE/ACM Transactions on Networking, Vol.10, No:4, 513-528.

[13] S.Athuraliya, et. al, “An Enhanced random early marking algorithm for internet flow control”, Proceedings of Infocom, Isreal, 2000