Active queue management (AQM) algorithms control congestion by controlling flow. Congestion is measured and a control action is taken. There are mainly two approaches for measuring congestion: (1) queue based, and (2) flow based. In queue based AQMs congestion is observed by queue size. The drawback of this is that a backlog of packets is inherently necessitated by the control mechanism, as congestion is observed when the queue is already positive. This creates unnecessary delay and jitter (delay variation). Flow based AQMs, on the other hand, determine congestion and take action based on the packet arrival rate. For such schemes, backlog, and all its adverse implications, is not necessary for the control mechanism.

Goals of an active queue management mechanism can be summarized as follows:

- Reducing number of packets dropped in routers: Keep average queue size small, hence leaving enough space for bursts.
- Providing lower-delay interactive service: By keeping average queue size small, end-to-end delays will be shorter.
- Avoiding lock-out behavior: Avoid bias against low bandwidth and bursty flows. Guarantee that a newly arriving packet ‘almost always’ finds a place in the buffer.

Active Queue Management (AQM) is an advanced form of router queue management that tries to detect and react to the congestion prior to its fatal consequences such as full queues and bursty drops. In reaction to suspected congestion, AQM schemes drop packets early (or do ECN-marking) to signal the congestion to the end nodes. The most important difference between the various AQM schemes is that when they suspect congestion, and how do they select the packets to be marked/dropped. In general, the congestion judgment can be based on current or averaged queue on the traffic’s arriving rate being higher than the departure rate, or other characteristics of the traffic or the queuing system, such as the number of recent tail-drops. According to the type of metrics used to measure congestion, AQM schemes can be classified into three catalogs: Queue-based, rate-based, and schemes based on concurrent queue and rate metrics. In queue-based schemes, congestion is observed by average or instantaneous queue length and the control aim is to stabilize the queue length. The drawback of queue-based schemes is that a backlog is inherently necessitated. Rate-based schemes predict the utilization of the link, determine the level of congestion, and take actions based on the packet arrival rate. Rate-based schemes can provide early feedback for congestion [3]. Other AQM schemes deploy a combination of queue length and input rate to measure congestion and achieve a tradeoff between queue stability and responsiveness. Current TCP/AQM algorithms assume that packet loss is mainly due to network congestion. However, these TCP/AQM algorithms are insufficient for the hybrid wired and wireless networks. In wireless networks, communication links have intrinsic characteristics that affect the performance of transport protocols including variable bandwidth, corruption, channel allocation delays, and asymmetry [5]. All of these cause significant no congestion packet loss. The generic TCP protocol notifies sources to reduce their transmission rates only when facing packet losses due to network congestion. Therefore, the TCP protocol in hybrid wired and wireless networks needs not only to detect packet losses, but also to detect the reason of packet losses. Moreover, the degree of statistical multiplexing number of flows over wireless links is different from that over wired links.
II. TAIL-DROP

The traditional technique for managing router queue lengths is to set a maximum length (in terms of packets) for each queue, accept packets for the queue until the maximum length is reached, then reject (drop) subsequent incoming packets until the queue decreases because a packet from the queue has been transmitted.

The tail-drop algorithm [7] was not designed to be an efficient AQM. It is simply a queue which, when filled to its maximum, overflows and drops any subsequently arriving packets. However, we can interpret it as an AQM, which measures the backlog to determine the congestion level. No congestion is detected by the tail-drop algorithm until the queue is full. As shown in Fig. 1, when the queue is full, the maximum congestion signal is generated because all of the arriving packets are dropped. Once sources detect lost packets they slow down and the arrival rate of packets to the queue will be less than the capacity of the link and the packet backlog in the queue decreases. Then, when the buffer is not full, no congestion feedback signal is generated by tail-drop algorithm [11] and the source rates increase until overflow happens again. We can see that the tail-drop AQM results in a cycle of decrease and increase of rates around the point where the buffer is nearly full. The actual mean size of the buffer depends on the load on the link. The tail-drop algorithm is incapable of generating any feedback signal (price) unless the buffer is full. This is why the current Internet suffers long queuing delays and long RTTs. Most work on Active Queue Management uses the tail-drop queue as a lower bound for performance comparison. However, in a basic mechanism is used to improve the fairness.

![Fig. 1 Tail-drop dropping probability vs queue Size](image)

III. RANDOM EARLY DETECTION

The most well-known AQM algorithm is the Random Early Detection algorithm [6] developed by Sally Floyd and Van Jacobson in 1993. The design of RED was a response to some of the problems with tail-drop queue. The Internet Engineering Task Force recommends the deployment of RED in RFC 2309 and indeed it is widely implemented in routers today. However, by and large it has not been switched on, and doubt has arisen about the degree to which it can improve network performance. In this section, we will describe RED and discuss some of the problems with the RED paradigm. RED is a descendent of the DECbit congestion avoidance scheme. RED uses queue length as a congestion measure. The rate of congestion notifications generated is directly related to the time averaged queue occupancy. The congestions notifications are either packet drops or ECN marks if the packet is ECN capable. The time-averaged queue occupancy is computed using an exponential averaging scheme, where $w_q$ is an averaging time constant, and $B(t)$ is the instantaneous queue occupancy:

$$\text{Avg}(t + 1) = (1 - w_q) \times \text{Avg}(t) + w_q \times B(t)$$

When the averaged queue length exceeds a minimum threshold, congestion notifications are generated with a probability which is proportional to the excess queue length as shown in Fig. 2. The parameter $min_{th}$ determines the queue length at which congestion notification begins and $max_{th}$ determines the point where congestion notification is performed on all packets. When the queue size exceeds the maximum queue length $B_{max}$, all packets are dropped.

![Fig. 2 RED dropping probability vs queue Size](image)

RED addresses a number of problems that tail-drop exhibits. When a tail-drop queue overflows, there is a large number of packets drops all at once. This will result in the global synchronisation problem. This problem has been reported by Floyd and Hashem. Global synchronisation is a cycle of under-utilisation following the burst of drops followed by a period of overload. It occurs because the burst of drops results in a large number of TCP sources reducing their window size at the same time. With RED, the early packet drops are randomised and spread over time. Packet
marking/dropping is uniformly random for each packet. This reduces global synchronisation of sources. Another problem with the tail-drop queue is Lock-Out. Lock-Out describes the effect when new TCP sessions receive significantly less bandwidth than established connections. This phenomenon is reported and stems from synchronisation phenomena. A key aim of RED is to reduce the average queue length to accommodate short bursts in the queue. RED achieves a lower average queue size than tail-drop because it generates congestion notifications once the queue is greater than $\min_{\text{th}}$ unlike tail-drop which drops packets once the queue size is greater than $B_{\text{max}}$. Despite some advantages of the RED AQMs over the tail-drop queue, RED exhibits a number of serious problems. It performs a control-theoretic analysis of TCP/RED and uncovers that this flow control mechanism eventually becomes unstable as RTT delay increases, or when network capacity is increased. In the following chapter, we will review recent work which analyses how stable control can be achieved for any number of flows, network delays, and capacities. The instability of RED is discussed in numerous papers. These argue that the jitter that RED[8] introduces demands such large jitter buffers of CBR applications (eg. audio streaming) that any delay performance improvements of RED are negated by the latency these jitter buffers introduce. In the difficulty of tuning RED parameters to observe performance increase in web traffic is discussed. The oscillatory behaviour and possible instability of RED is again confirmed by a more analytical approach. In “Reasons not to deploy RED” it is questioned whether RED gives any benefit over tail-drop. The RED algorithm is a basis for many other AQM proposals which seek to enhance the performance of the basic RED structure. For example, in a method to balance bandwidth among flows more equally is introduced. The proposal called “RED+” adds the ability to identify and discriminate against high-bandwidth, “unfriendly” flows, which do not implement congestion control. The variant “SRED”, Stabilised RED, presented, tunes RED’s parameters by estimating the number of connections present. Further, we will show that measuring the queue occupancy to determine congestion is a fundamental structural constraint of RED and RED-like AQMs. Rather than continuing to enhance the RED AQM structure, we will investigate structurally different paradigms for AQM design. We will show that new AQMs whose structure is based on packet arrival rate measurement, remove the limitations and performance issues of the RED-like AQMs. We will also develop techniques for applying these rate-based AQMs in applications where RED AQMs were previously suggested.

IV. RANDOM EXPONENTIAL MARKING

REM [1] is both a set of AQMs and a novel technique for communicating congestion information. REM embodies a mechanism for the precise communication of link congestion prices, so that the link congestion state variable is exactly the congestion price as in the utility optimization. A REM link marks a packet at link 1 with a probability based on the link price pl state, and a global encoding constant $\phi$ (1 < $\phi$):

$$M(t) = 1 - \phi^n$$ (1)

Because sources know the value of $\phi$, they can compute the total end-to-end path congestion price. Therefore, in a complete deployment, REM requires a REM link algorithm and a source algorithm capable 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 $p_l(t)$ state variable can be interpreted as the marking rate, just as the other AQMs discussed. Indeed for $p_l(t) < 1$, we can assume $p_l(t)=m(t)$ by (1). With this in mind, the three control laws PC1, PC2, PC3 can be interpreted as alternative AQMs:

$$P(t+1) = \gamma b(t)$$ (2)

$$P(t+1) = [p(t) - \gamma(x(t) - c)]^+$$ (3)

$$P(t+1) = [p(t) - \gamma(x(t) + b(t) - c)]^+$$ (4)

Where $p(t)$ is the congestion notification rate, $c$ is a target capacity just below the actual link capacity, $b(t)$ is the backlog, and $\gamma$ and $\alpha$ are control constants which affect speed and stability of control.

It is clear that PC1 control law is very similar to the RED-like AQMs where the congestion notification rate is proportional to backlog. It is the PC2 and PC3 control laws that present a new approach to AQM design. PC2 and PC3 uncouple the congestion notification rate from the backlog at the link. PC2 and PC3 measure the arrival rate to the link to compute the congestion notification rate instead of using the backlog. The congestion notification rate is controlled by an integral controller, whose error term is the discrepancy between the aggregate 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, which makes the marking rate increase with greater rate if there is a backlog. This was found to improve the transient response of the basic PC2 controller, and reduce the amount of backlog during transient periods when the load changes. The stability properties of PC3 are analysed in. Further work by S. Low and co-authors in this area extends the analysis and improves the REM framework. Several of his 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, the buffer requirements, and jitter are reduced. An extension to the control equation based on a Newton-like algorithm is analysed. An approach using a deadbeat controller is used. Experimental results indicate that the control laws PC2-PC3 are able to control the sources such so that the mean backlog at the link is substantially reduced compared to a tail-drop queue or RED. The result indicates that these AQMs are able to operate with very low backlog, and maintain a high link utilisation. The PC2-PC3 depart significantly both in performance and design from the RED or tail-drop algorithm.

V. GREEN AQM

In this section, we introduce a new RB AQM algorithm called GREEN. GREEN [12] is an extension of the basic integrator AQM control law. The GREEN algorithm is described by the following control law:

© 2013, IJARCSSE All Rights Reserved
\[ P(t + \Delta T) = P(t) + \varphi(t) \cdot U(\delta(t)) \]

where

\[ \delta(t) = \Pi(X(t) - u \cdot C(t)) \]

\[ U(x) = \begin{cases} +1 & x \geq 0 \\ -1 & x < 0 \end{cases} \]

\[ \varphi(t) = \max(\text{abs}(\delta(t)), \Delta P) \]

where \( X(t) \) is the link’s estimated arrival rate (bps), \( C(t) \) is the link capacity, \( u \) is the target utilisation, \( \alpha \) is the control gain, \( P(t) \) is the marking/dropping rate, \( \Delta P \) determines the minimum adjustment, and \( 1/\Delta T \) is the update rate. To visualise the GREEN algorithm, the amount of adjustment to \( P(t) \), \( \delta P(t) \) for every \( \Delta T \) seconds, as a function of \( t \) is shown for GREEN and a linear integrator control law in Fig. 3.

![Fig. 3 (left) GREEN P(t) adjustment(right) linear integrator P(t) adjustment.](image)

The target link utilisation \( u \) is assigned a value less than 1 to clear the link buffer in equilibrium. The control gain \( \alpha \) and the minimum adjustment \( \Delta P \) should be scaled in proportion to \( 1/C \), the expected link capacity, to make stability invariant to link capacity. The update interval \( \Delta T \) should be an order less than the shortest RTT in the network.

It is known that the convergence rate of the integrator controller is not optimal. It is important for the arrival rate to a link to be controlled towards the target capacity quickly, since long transient periods, where arrival rate is far below capacity, create under-utilisation and periods where arrival rate is above capacity create queuing and delay. GREEN improves the integral control law by a first-order method of limiting the minimum adjustment of the marking probability \( P(t) \) to \( \Delta P \) per update interval. Another method of accelerating the integrator requires the computation of a scaling matrix which involves division, an operation not readily amenable to FPGA or ASIC implementation with limited space and time resources.

VI. SIMULATION RESULTS

The AQMs were tested under different traffic loads, by varying the number of active saturated TCP sessions. The total number of TCP sessions activated in each 120s simulation period ranged from 100 to 1300 (mean of 11 to 141 concurrent sessions). The buffer size was 91 packets (72.8ms), so that it is lower than the bandwidth delay product of some connections. The link utilisation, (i.e., percentage of time the 1-2 link is not idle), packet loss rate (at 1-2 link due to buffer overflow), and queuing delay at the link were measured. Table 1. Summarizes the mean of utilisation, loss and delay over the 13 simulation periods for each AQM. Fig. 4. Shows the packet delay performance and Fig 5. Shows the packet loss performance across the range of loads for all AQMs.

**TABLE I: TRIAL 1 RESULT SUMMARY-MEAN UTILIZATION LOSS DELAY FOR AQMS**

<table>
<thead>
<tr>
<th></th>
<th>Tail-drop</th>
<th>RED</th>
<th>REM</th>
<th>GREEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Util%</td>
<td>0.98</td>
<td>0.98</td>
<td>0.96</td>
<td>0.97</td>
</tr>
<tr>
<td>Loss%</td>
<td>6.85</td>
<td>5.45</td>
<td>0.23</td>
<td>0.05</td>
</tr>
<tr>
<td>Delay(ms)</td>
<td>52.45</td>
<td>52.4</td>
<td>21.68</td>
<td>18.79</td>
</tr>
</tbody>
</table>

![Fig. 4 Mean queueing delay vs number of TCP sessions](image)
Fig. 4 Shows that the RB AQMs REM and GREEN both exhibited low queuing delay across all loads. This resulted in almost no packet loss as shown in Fig 5. Notice from Table 1, that the 2 to 3 times lower queuing delay of RB AQMs as compared to the BB AQM occurs with only a 1-2% decrease in utilisation. This is consistent with basic queuing theory results for queuing with random arrival processes.

For the BB AQMs, RED and tail-drop, the delay (backlog) increased with the load. The delay necessarily increases with the load because of the coupled relationship between backlog \( B(t) \) and congestion notification rate \( P(t) \), \( P(t)=f(B(t)) \). This also explains the increased loss rate with load, since an increasing mean queue size increases the probability of loss. Because of the coupling between \( P(t) \) and \( B(t) \), the BB AQMs RED and tail-drop cannot target an arbitrary utilisation level, since \( B(t) \) must be positive and increasing with load. As the load increases, these AQMs inevitably drive the system at higher utilisation and result in higher backlog than RB AQMs. RED has a slightly lower loss rate than tail-drop, because it is able to mark some packets using ECN, rather than having the entire congestion notification signal being packet drops.

VII. CONCLUSIONS

In this paper, we have explained the main goals of AQM. In this work, the performance of AQM schemes has been evaluated. We have compared Drop tail, RED, REM, and GREEN algorithms. It has been demonstrated the effectiveness and weakness of these algorithms. AQM algorithms are absolutely useful because the management of packets to avoid congestion occasionally requires exceeding hardware capabilities. The deployment of modern AQM algorithms in switches and routers will improve the delay performance for all applications using the network, and will enable multimedia and interactive applications to coexist on the best-effort network.

REFERENCES


