İstanbul Ticaret Üniversitesi Fen Bilimleri Dergisi Yıl:4 Sayı:7 Bahar 2005/1 s.123-140

EVALUATION OF ACTIVE QUEUE MANAGEMENT ALGORITHMS

Serhat ÖZEKES[*]

ABSTRACT
Active Queue Management (AQM) is a very active research area in networking flows. In order to stem the increasing packet loss rates caused by an exponential increase in network traffic, researchers have been considering the deployment of active queue management algorithms. In this paper we will evaluate the active queue management algorithms such as Droptail, RED, BLUE, REM, GREEN and PURPLE. The algorithms selected from amongst the many published over the past ten years, will be described in a simplified manner. Strengthness and weakness of these algorithms will be discussed.

Keywords: AQM, QoS, ECN, Droptail, RED, BLUE, REM, GREEN, PURPLE

AKTİF KUYRUK YÖNETİM ALGORİTMALARININ DEĞERLENDİRİLMESİ

ÖZET
Aktif kuyruk yönetimi, ağ trafiğinin düzenlenmesinde güncel bir araştırma konusudur. Ağ trafiğindeki paket kayıplarının artışının önlenmesi için araştırmacılar çalışmalarında aktif kuyruk yönetim algoritmalarına ağırlık vermektedirler. Bu çalışmada Droptail, RED, BLUE, REM, GREEN ve PURPLE gibi aktif kuyruk yönetim algoritmaları değerlendirilecektir. Bu konu üzerinde yapılmış çalışmalardan seçilen algoritmalar özet bir şekilde açıklanacaktır. Bu algoritmaların güçlü ve zayıf yanları tartışılacaktır.

Anahtar Kelimeler: AQM, QoS, ECN, Droptail, RED, BLUE, REM, GREEN, PURPLE

[*] *Istanbul Commerce University, Vocational School, Uskudar – ISTANBUL, serhat@iticu.edu.tr*

1. INTRODUCTION

Increasing access to data communications is creating sweeping changes around the globe. More people are using a wider range of services, requiring more data to be transported. As a result, there has been a surge of interest in designing low-loss and low-delay networks by encouraging users to adapt to changing networks conditions using minimal information from the network.

Ultimately the performance of a communications network will be judged by the Quality of Service (QoS) perceived by users. This end user QoS can be affected by many factors outside of the control of the network operators. For example, the quality of an MPEG-1 video stream, which has very limited error suppression capabilities, will be perceptibly lower than that of a video service with error suppression and correction capabilities built-in after both streams have been transported across an inherently lossy network like the Internet. The differences in the perceived quality will obviously be affected by the performance of the transport network, but the main differences will come from the nature of the service being transported (Neame, 2003).

Aggregate queuing delay or latency is the amount of time it takes the senders packet, once it enters the network, to be delivered to its destination. Figure 1 shows a single sender-receiver connection. A packet is a form of data in a sequence of binary digits in a packet-switched network. Transmission Control Protocol (TCP) divides a file into efficient sized packets that are separately numbered. A simple packet contains an IP header, packet number protocol, source address (TCP sender), destination Internet address, and data. The TCP sends the packet into the network where it is then routed through various links and hubs. Each link is connected to a hub server that routes packets through routers and switches that have a queue size and possibly a different algorithm for handling queue congestion. The hub server's router and switches take input links and route the packets to the appropriate output link. When a packet enters a hub server, it will be placed into a routers queue when there is congestion. If there is no congestion the packet will be sent immediately with virtually zero delay. Depending on the length of the queue and where the packet was placed in the queue determines the amount of the time it will reside in queue until it is sent. Once all the packets have arrived to its destination, the TCP receiver will reassemble the file in the receiver TCP by putting the packets in-order and assembling the data back into a file (Manley, 2003).

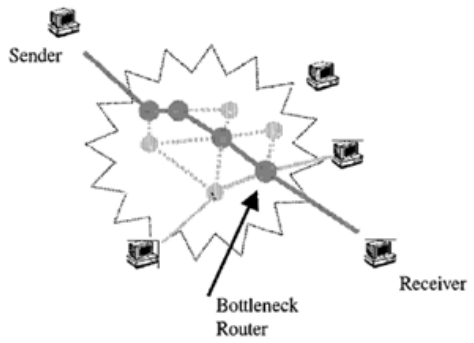

Figure 1. A single sender-receiver connection (Hollot, 2002).

Current versions of TCP rely on loss as indicators of congestion. Clearly, this is undesirable if one wants to operate the network at low-levels of loss. On the other hand, losses are good indicators of congestion and one needs other signals from the network to make congestion-control decisions with very little or no loss. The schematic of a sender receiver connection is seen in Figure 2. *Explicit Congestion Notification* (ECN) has been recently proposed to provide early indication to sources about imminent congestion in the network. ECN marking is a mechanism to provide such information about the network to the users.

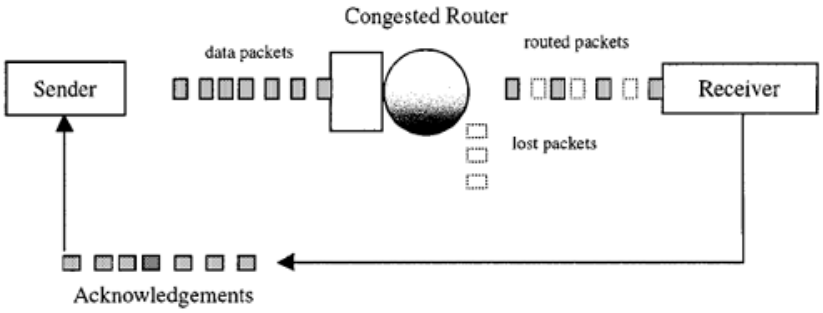

Figure 2. A schematic of a sender-receiver connection (Hollot, 2002).

To provide ECN marks, the routers need to mark packets intelligently that conveys information about the current state of the network to the users. Algorithms which the routers employ to convey such information are called as *Active Queue Management* (AQM) schemes. There are various AQM schemes that have proposed in the literature which are summarized in Figure 3. Here are the currently proposed AQM algorithms (Hollot, 2002):

- Drop Tail
- RED
- BLUE
- REM
- GREEN
- PURPLE

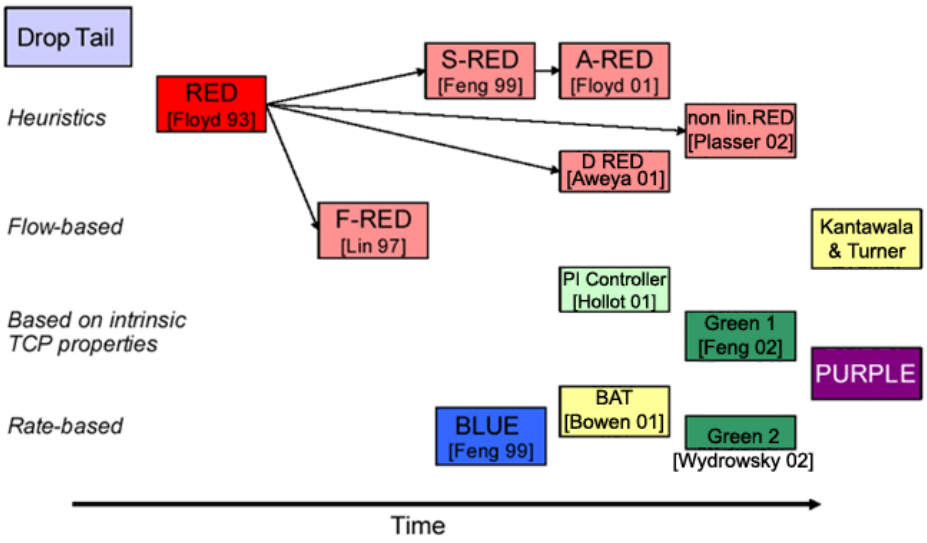

Figure 3. The development of AQM algorithms by time (Pletka et al., 2003/a).

In general, AQM schemes 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 (Wydrowski and Zukerman, 2002).

Goals of an active queue management mechanism can be summarized as follows (Braden et al., 1998):

1. Reducing number of packets dropped in routers: Keep average queue size small, hence leaving enough space for bursts.

2. Providing lower-delay interactive service: By keeping average queue size small, end-to-end delays will be shorter.

3. 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.

This paper will discuss the above proposed AQM algorithms. There will be comparisons among these algorithms. In section 2 Drop Tail, in section 3 RED, in section 4 BLUE, in section 5 REM, in section 6 GREEN, in section 7 PURPLE will be discussed and section 8 will conclude the paper.

2. DROP TAIL

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. This technique is known as 'drop tail', since the packet that arrived most recently (i.e., the one on the tail

of the queue) is dropped when the queue is full. This method has served the Internet well for years, but it has two important drawbacks (Braden et al., 1998):

1. Lock-Out: In some situations drop tail allows a single connection or a few flows to monopolize 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.

2. 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 steady-state queue size, and this is perhaps queue management's most important goal.

In short, drop tail is effectively 'no management'. As the demand on networks increased, the amount of data being passed through links and hubs could no longer be unmanaged. A new algorithm was developed called Random Early Detection, the first real AQM algorithm to manage data congestion (Wydrowski and Zukerman, 2002).

3. THE RED (RANDOM EARLY DETECTION) ALGORITHM

One of the biggest problems with TCP's congestion control algorithm over drop tail queues is that sources reduce their transmission rates only after detecting packet loss due to queue overflow. Since a considerable amount of time may elapse between the packet drop at the router and its detection at the source, a large number of packets may be dropped as the senders continue transmission at a rate that the network cannot support. RED alleviates this problem by detecting incipient congestion *early* and delivering congestion notification to the end-hosts, allowing them to reduce their transmission rates before queue over- flow occurs.

Van Jacobson and Sally Floyed first introduced the RED algorithm in August of 1993 (Floyd and Jacobson, 1993; Feng et al., 1999/a). RED has been designed with the objective to minimize packet loss and queuing delay, avoid global synchronization of sources, maintain high link utilization and remove biases against bursty sources.

The congestion scenario presented in Figure 4 occurs when a large number of TCP sources are active and when a small amount of buffer space is used at the bottleneck link. As the Figure shows, at t = 1, a sufficient change in aggregate TCP load (due to TCP opening its congestion window) causes the transmission rates of the TCP sources to exceed the capacity of the bottleneck link. At t = 2, the mismatch between load and capacity causes a queue to build up at the bottleneck. At t = 3, the average queue length exceeds minth and the congestion-control mechanisms are triggered. At this point, congestion notification is sent back to the end hosts at a rate dependent on the queue length and marking probability maxp. At t = 4, the TCP receivers either detect packet loss or observe packets with their ECN bits set. In response, duplicate acknowlegdements and/or TCP-based ECN signals are sent back to the sources. At t = 5, the duplicate acknowlegements and/or ECN signals make their way back to the sources to signal congestion. At t = 6, the sources finally detect congestion and adjust their transmission rates. Finally, at t = 7, a decrease in offered load at the bottleneck link is observed. Note that it has taken from t = 1 until t = 7 before the offered load becomes less than the link's capacity. Depending upon the aggressiveness of the aggregate TCP sources and the amount of buffer space available in the bottleneck link, a large amount of packet loss and/or deterministic ECN marking may occur. Such behavior leads to eventual underutilization of the bottleneck link (Feng, 2002/b).

One way to solve this problem is to use a large amount of buffer space at the RED gateways. For example, it has been suggested that in order for RED to work well, an intermediate router requires buffer space that amounts to twice the bandwidth-delay product. This approach, in fact, has been taken by an increasingly large number of router vendors. Unfortunately, in networks with large bandwidth-delay products, the use of large amounts of buffer adds considerable end-to-end delay and delay jitter. This severely impairs the ability to run interactive applications. In addition, the abundance of deployed routers which have limited memory resources makes this solution undesirable (Feng, 2002/b; Lin and Morris, 1997).

Figure 5 shows how an ideal queue management algorithm works. In this Figure, the congested gateway delivers congestion notification at a rate which keeps the aggregate transmission rates of the TCP sources at or just below the clearing rate. While RED can achieve this ideal operating point, it can do so only when it has a sufficiently large amount of buffer space and is correctly parameterized (Feng, 2002/b).

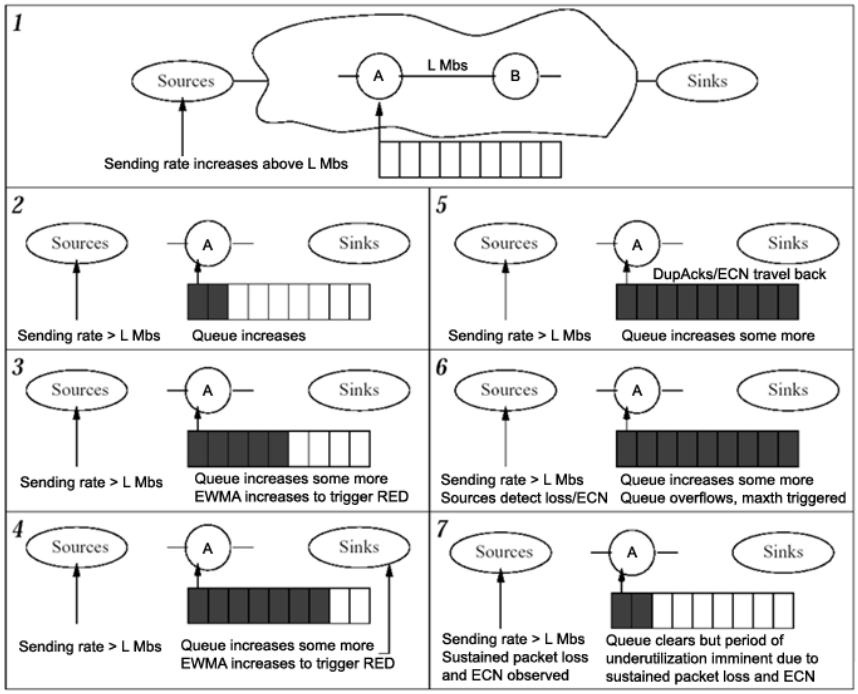

Figure 4. RED example (Feng, 2002/b)

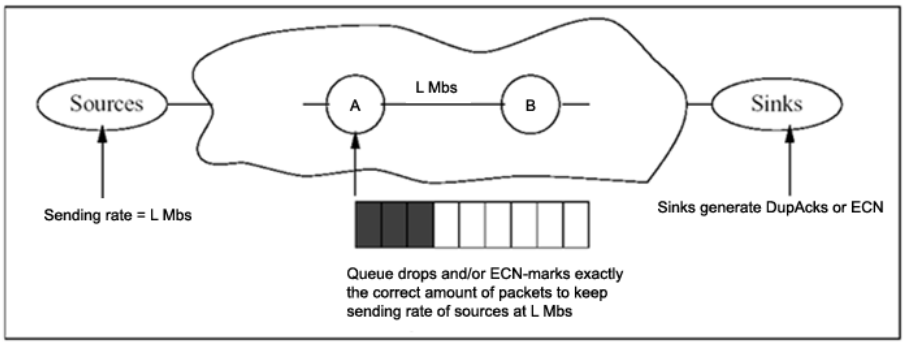

Figure 5. Ideal scenario (Feng, 2002/b)

RED gateway calculates the average queue size, using a low-pass filter with an exponential weighted moving average. The average queue size is compared to two thresholds, a *minimum* threshold and a *maximum* threshold (Floyd and Jacobson, 1993). When the average queue size is less than the minimum threshold, no packets are marked. When the average queue size is greater than the maximum threshold, every arriving packet is marked. If marked packets are in fact dropped, or if all source nodes are cooperative, this ensures that the average queue size does not significantly exceed the maximum threshold (Floyd and Jacobson, 1993; Lin and Morris, 1997).

When the average queue size is between the minimum and the maximum threshold, each arriving packet is marked with probability *pa*, where *pa* is a function of the average queue size *avg*. Each time that a packet is marked, the probability that a packet is marked from a particular connection is roughly proportional to that connection's share of the bandwidth at the gateway. The general RED gateway algorithm is given in Figure 6 (Floyd and Jacobson, 1993).

```
for each packet arrival
    calculate the average queue size avg
    if minth avg < maxth
        calculate probability pa
        with probability pa:
            mark the arriving packet
    else if maxth avg
        mark the arriving packet
```

Figure 6. General algorithm for RED gateways (Floyd and Jacobson, 1993).

RED interacts with TCP: as source rates increase, queue length grows, more packets are marked, prompting the sources to reduce their rates, and the cycle repeats. TCP defines precisely how the source rates are adjusted while active queue management defines how

the congestion measure is updated. For RED, the congestion measure is queue length and it is *automatically* updated by the buffer process. The queue length in the next period equals the current queue length plus aggregate input minus output (Athuraliya et al., 2001)

$$b_l(t+1) = \left[b_l(t) + x_l(t) - c_l(t)\right]^+ \tag{1}$$

where $[z]^+ = \max\{z,0\}$. Here $b_l(t)$ is the aggregate queue length at queue l in period t, $x_l(t)$ is the aggregate input rate to queue l in period t, and $c_l(t)$ is the output rate in period t (Athuraliya et al., 2001).

4. THE BLUE ALGORITHM

The BLUE algorithm resolves some of the problems of RED by employing the use of a hybrid flow control scheme along with a queue size congestion measuring scheme. It uses flow and queue events to modify the congestion notification rate. This rate is adjusted by two factors: packet loss from queue congestion and link utilization or underutilization. A key difference the BLUE algorithm has from RED, is that uses packet loss rather than the average queue length (Feng, 1999/b).

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

The typical parameters of BLUE are *d1*, *d2*, and *freeze time*. *d1* determines the amount by which *Pm* is increased when the queue overflows, while *d2* determines the amount by which *Pm* is decreased when the link is idle. f*reeze time* is an important parameter that determines the minimum time interval between two successive updates of *Pm*. 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 Figure 7 (Feng, 2002/b):

Upon link idle event:	**Upon packet loss event:**
if ((now-last update)>freeze time) Pm = Pm-d2; Last update = now;	if ((now–last update)>freeze time) Pm = Pm+d1; Last update = now;

Figure 7. The BLUE algorithm (Feng, 2002/b).

Here are some problems with BLUE (Feng, 1999/b):

- BLUE uses a hashing function to discover the non-responsive flows. This depends on the assumption that non-responsive flows would not be very large in number. We know that this is true but there could be cases when this assumption does not hold true.
- When we have large number of non-responsive flows, they could pollute the bins and TCP flows could be mistaken to be non-responsive, resulting in needlessly penalizing them.
- One solution to this could be that we change the hash functions during regular intervals of time. This would map some responsive flows to unpolluted bins.
- Another problem is that once a flow is marked, it is tainted for ever. If later the flow restrains itself, BLUE still tries to reduce its sending rate through packet drops.

5. THE REM (RANDOM EXPONENTIAL MARKING) ALGORITHM

REM aims to achieve both high utilization and negligible loss and delay in a simple and scalable manner. The key idea in achieving this is to decouple congestion measure from performance measure such as loss, queue length or delay. While congestion measure indicates excess demand for bandwidth and must track the number of users, performance measure should be stabilized around their targets independently of the number of users.

REM that has the following key features (Athuraliya et al., 2001):

1. match rate clear buffer:
It attempts to match user rates to network capacity while clearing buffers (or stabilize queues around a small target), regardless of the number of users.

2. sum prices:
The *end-to-end* marking (or dropping) probability observed by a user depends in a simple and precise manner on the *sum* of link prices (congestion measures), summed over all the routers in the path of the user.

The 'match rate clear buffer' feature implies that, contrary to the conventional wisdom, high utilization is not achieved by keeping large backlogs in the network, but by feeding back the right information for users to set their rates. The first idea of REM is to both stabilize the input rate around link capacity and the queue around a small target, *regardless of the number of sources sharing the link*. Each output queue that implements REM maintains a variable called 'price' as a congestion measure. This variable is used to determine the marking probability, as explained in the next subsection. Price is updated, periodically or asynchronously, based on rate mismatch (i.e., difference between input rate and link capacity) and queue mismatch (i.e., difference between queue length and target). The price is incremented if the weighted sum of these mismatches is positive, and decremented otherwise. The weighted sum is positive when either the input rate exceeds the link capacity or there is excess backlog to be cleared, and negative otherwise. When the number of sources increases, the mismatches in rate and in queue grow, pushing up price and hence marking probability. This sends a stronger congestion signal to the sources which then reduce their rates. When the source rates are too small, the mismatches will be negative, pushing down price and marking probability and raising source rates, until eventually, the mismatches are driven to zero, yielding high utilization and negligible loss and delay in equilibrium. The buffer will be cleared in equilibrium if the target queue is set to zero (Athuraliya et al., 2001).

Whereas the congestion measure (queue length) in RED is automatically updated by the buffer process according to (1), REM *explicitly* controls the update of its price to bring about its first property. Precisely, for queue l, the price $p_l(t)$ in period t is updated according to (Athuraliya et al., 2001):

$$p_l(t+1) = \left[p_l(t) + \gamma(\alpha_l(b_l(t) - b_l^*) + x_l(t) - c_l(t))\right]^+ \quad (2)$$

where $\gamma>0$ and $\alpha_l>0$ are small constants and $[z]^+ = \max\{z,0\}$. Here, $b_l(t)$ is the aggregate buffer occupancy at queue l in period t and $b_l^* \geq 0$ is target queue length, $x_l(t)$ is the *aggregate* input rate to queue l in period t , and $c_l(t)$ is the available bandwidth to queue l in period t . The difference $x_l(t)$-$c_l(t)$ measures rate mismatch and the difference $b_l(t)$-b_l^* measures queue mismatch. The constant α_l can be set by each queue individually and trades off utilization and queueing delay during transient. The constant γ controls the responsiveness of REM to changes in network conditions. Hence, from (2), the price is increased if the weighted sum of rate and queue mismatches, weighted by α_l , is positive, and decreased otherwise. In equilibrium, the price stabilizes and this weighted sum must be zero. i.e., $\alpha_l(b_l-b_l^*)+x_l-c_l=0$. This can hold only if the input rate equals capacity ($x_l=c_l$) and the backlog equals its target ($b_l=b_l^*$), leading to the first feature mentioned at the beginning (Athuraliya et al., 2001).

The 'sum prices' feature is essential in a network where users typically go through multiple congested links. It clarifies the meaning of the congestion information embedded in the end-to-end marking (or dropping) probability observed by a user, and thus can be used to design its rate adaptation (Athuraliya et al., 2001).

The output queue marks each arrival packet that is not already marked at an upstream queue, with a probability that is exponentially increasing in the current price. The exponential form of the marking probability is critical in a large network where the end-to-end marking probability for a packet that traverses multiple congested links from source to destination depends on the link marking probability at every link in the path. When, and only when, individual link marking probability is exponential in its link price, this end-to-end marking probability will be exponentially increasing in the *sum* of the link prices at all the congested links in its path. This sum is a precise measure of congestion in the path. Since it is embedded in the end-to-end marking probability, at every link in the path. When, and only when, individual link marking probability is exponential in its link price, this end-to-end marking probability will be exponentially increasing in the *sum* of the link prices at all the congested links in its path. This sum is a precise measure of congestion in the path. Since it is embedded in the end-to-end marking probability, it can be easily estimated by sources from the fraction of their own

packets that are marked, and used to design their rate adaptation (Athuraliya et al., 2001).

Precisely, suppose a packet traverses links $l=1,2,...,L$ that have prices $p_l(t)$ in period t. Then the marking probability $m_l(t)$ at queue l in period t is (Athuraliya et al., 2001):

$$m_l(t) = 1 - \phi^{-p_l(t)} \tag{3}$$

where $\phi > 1$ is a constant. The end-to-end marking probability for the packet is then:

$$1 - \prod_{l=1}^{L} (1 - m_l(t)) = 1 - \phi^{-\Sigma_l p_l(t)} \tag{4}$$

i.e., the end–to–end marking probability is high when the congestion measure of its path, $-\Sigma_l p_l(t)$, is large.

When the link marking probabilities $m_l(t)$ are small, and hence the link prices $p_l(t)$ are small, the end-to-end marking probability given by (5) is approximately *proportional to* the sum of the link prices in the path (Athuraliya et al., 2001):

end-to-end marking probability $\cong (\log_e \phi) \sum_l p_l(t)$

6. THE GREEN (GENERALIZED RANDOM EARLY EVASION NETWORK) ALGORITHM

The GREEN Algorithm is a feedback control function which adjusts the rate of congestion notification in response to the flow based congestion measure, x_{est}, the estimated data arrival rate. GREEN is based on a threshold function. If the link's estimated data arrival rate x_{est} is above the target link capacity c_1, the rate of congestion notification, P, is incremented by ΔP at a rate of $1/\Delta T$. Conversely, if x_{est} is below c_l, P is decremented by ΔP at a rate of $1/\Delta T$. The algorithm applies probabilistic marking of incoming packets at the rate P, either by dropping packets or setting the ECN. Let the step function $U(x)$ be defined by (Wydrowski and Zukerman, 2002; Feng et al., 2002/a):

$$U(x) = \begin{cases} +1 & x \geq 0 \\ -1 & x < 0. \end{cases} \tag{5}$$

Therefore,

$$P = P + \Delta P.U(x_{est} - c_t) \tag{6}$$

The target link capacity c_t is assigned a value just below the actual capacity c, typically $0.97\,c$, so that the queue size converges to 0. Incoming data rate estimation is performed using exponential averaging:

$$x_{est} = (1 - \exp(-Del/K)) * (B/Del) + \exp(-Del/K) * x_{est} \tag{7}$$

where *Del* is the inter-packet delay, *B* the packet size and *K* the time constant. Other arrival rate estimation techniques could also be used successfully.

There is a relationship between REM and GREEN. If Eq (3) is linearised, $m = P$, the exponential term is eliminated. Furthermore if the buffer term $\alpha = 0$, and the linear constant γ is replaced with the step function (5), GREEN's congestion notification rate *P* becomes equivalent to REM's price P_1 (Wydrowski and Zukerman, 2002).

7. THE PURPLE ALGORITHM

The PURPLE approach, in contrast to others, predicts the impact of its own actions on the behavior of reactive protocols and thus the short-term future traffic (Pletka et al., 2003/b). PURPLE achieves this by analyzing end-to-end information about the congestion state in the network. PURPLE allows much faster convergence of the main AQM parameters, at least towards a local optimum, thereby smoothing and minimizing both congestion feedback and queue occupancy. To improve the prediction, in (Pletka et al., 2003/b) it is also passively monitored (using lightweight operations) information pertaining to the amount of congestion elsewhere in the network as seen by flows traversing this router.

PURPLE provides smooth packet marking rates and queuing delay without any tuning of parameters because of its online optimized, autonomous behavior that relies on online model-based predictions. It is also able to avoid taildrop losses almost completely while providing an excellent balance between goodput, throughput, and average delay. This is achieved using minimal effort and state information thanks to the introduction of three new mechanisms, namely end-to-end congestion analysis, monitoring of ECN information, and use of the TCP model equation. Using simulations, in (Pletka et al., 2003/b) it has been shown that PURPLE behaves very well in a variety of circumstances.

8. CONCLUSION

In this paper, we have mentioned the terms AQM (Active Queue Management), QoS (Quality of Service) and ECN (*Explicit Congestion Notification*). We have explained the main goals of AQM. In this work, the performance of six AQM schemes, selected from amongst the many published over the past ten years, has been evaluated. We have compared Droptail, RED, BLUE, REM, GREEN and PURPLE algorithms. It has been demonstrated strengthness and weakness of these 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.

REFERENCES

Athuraliya S., Lapsley D. E., Low S. H., (2001), “Random early marking for Internet congestion control”, IEEE/ACM Transactions on Networking, Vol. 15, No:3, 48-53

Braden B., Clark D., Crowcroft J., Davie B., Deering S., Estrin D., Floyd S., Jacobson V., Minshall G., Partridge C., Peterson L., Ramakrishnan K. K., Shenker S., and Wroclawski J., (1998), “Recommendations on queue management and congestion avoidance in the internet”, Internet Draft

Feng W., Kandlur D., Saha D., Shin K. , (1999/a), "A Self-Configuring RED Gateway", In Proc. IEEE INFOCOM, 1320–1328

Feng W., Kandlur D., Saha D., Shin K., (1999/b), "Blue: A new class of active queue management schemes," Technical Report, CSE-TR-387-99, U. Michigan 286-299

Feng W., Kapadia A., Thulasidasan S., (2002/a), "GREEN: Proactive Queue Management over a Best-Effort Network", In Proceedings of IEEE Global Telecommunications Conference (GLOBECOM 2002), Taipei, Taiwan, Vol.21, No:1, 1784-1788

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

Floyd S., Jacobson V., (1993), "Random early detection gateways for congestion avoidance", IEEE/ACM Trans. On Networking, Vol.1, No:4, 397–413

Hollot C. V., Misra V., Towsley D., Gong W., (2002), "Analysis and Design of Controllers for AQM Routers Supporting TCP Flows", IEEE Transactions on Automatic Control, Vol. 47, No: 6, 945-959

Lin D., Morris R., (1997), "Dynamics of Random Early Detection", In Proc. of ACM SIGCOMM, 127-137

Manley R., (2003), "Reducing Packet Loss and Latency by Active Queue Management Algorithms", CSci 3902 - Seminar I, UMM CSci Twiki

Neame T., (2003), "Characterisation and Modelling of Internet Traffic Streams", Ph. D. Thesis, Department of Electrical And Electronic Engineering, The University Of Melbourne, 1-3

Pletka R., Waldvogel M., Mannal S., (2003/a), "PURPLE: Predictive Active Queue Management Utilizing Congestion Information", Proceedings of the 28th Annual IEEE Conference on Local Computer Networks LCN 2003, 21-30

Pletka R., Waldvogel M., Mannal S., (2003/b), “PURPLE: Predictive Active Queue Management Utilizing Congestion Information”, Presentation Slide, LCN ’03, Bonn / Königswinter

Wydrowski B., Zukerman M., (2002), "GREEN: An Active Queue Management Algorithm for a Self Managed Internet", Proceedings of ICC 2002, New York, Vol. 4, 2368-2372