Modeling and Performance Analysis of Dynamic Random Early Detection (DRED) Gateway for Congestion Avoidance

A. A. Akintola, G. A. Aderounmu, and L. A. Akanbi
Obafemi Awolowo University, Ile-Ife, Nigeria

aakintola69@yahoo.com
gaderoun@yahoo.com
lukman_akanbi@yahoo.com

M. O. Adigun
University of Zululand, Kwadlangezwa, Republic of South Africa

madigun@pan.uzulu.ac.za

Abstract

One of the most prominent congestion avoidance schemes in the Internet architecture is the Random Early Detection (RED) algorithm. Several modifications and enhancements have been made to the original RED so as to make it more responsive to congestion avoidance at the gateways. In this paper, we introduced the Dynamic Random Early Detection (DRED) model, which uses a newly introduced parameter i.e. warning line. A robust and efficacious technique to measure the burstiness of incoming traffic has been developed and tested. This involves the estimation of the average queue size, avg, which is dynamically adjusted hence the name of our scheme. The empirical results obtained from the simulations show that our DRED scheme responds early enough to the increased number of packets at the gateway. Also, the maximum drop probability of packets show improved performance over the original RED. It was concluded that our scheme demonstrated superiority by avoiding global synchronization and there is great reduction in the fluctuations of the actual queue size. Also, its early response avoids buffer overflow at the gateways when the queue is near full.

Keywords: tcp, congestion, DRED, gateway, queues, packet drop, fairness, throughput, probability

Introduction

Computer networks have experienced an explosive growth over the past few years and with that growth have come severe congestion problems. For example, it is now common to see Internet gateways to drop 10% of the incoming packets because of local buffer overflows. Investigations of some of these problems have shown that much of the course lies in transport protocol implementation (not in the protocols themselves). The obvious ways to implement a window-based transport protocol can result in exactly the wrong behaviour in response to network congestion (Jacobson, 1998). In 1993, Floyds and Jacobson presented a very interesting work about how to detect congestion using routers provided by a

random early detection mechanism. In high speed networks with connection with are likely to be designed with correspondingly large maximum queues to accommodate transient congestion. In the current Internet, the TCP (Transmission Control Protocol) transport protocol detects congestion only after a packet has been dropped at the gateway. However, according to Floyds and Jacobson (1993), it would be clearly undesirable to have large queues (possibly on the order of a delay-bandwidth product) that were near full much of the time; this would significantly increase the average delay in he network. Therefore, with increasingly high-speed networks, it is increasingly important to have mechanisms that keep throughput high but average queue sizes low.

In this paper, the authors are interested in the concept of Dynamic Random Early Detection (DRED) Gateway for congestion avoidance. In the absence of explicit feedback from the gateway, transport-layer protocols could infer congestion from the estimated bottleneck service time, from changes in throughput, from changes in end-to-end delay, as well as from packet drops or other methods. Only the gateway has a unified view of the queueing behaviour over time; the perspective of individual connections is limited by the packet arrival patterns for those connections. In addition, a gateway is shared by many active connections with a wide range of roundtrip times, tolerances of delay, throughput requirements, etc; decisions about the duration and magnitude of transient congestion to be allowed at the gateway are best made by the gateway itself.

Congestion Control and Resource Allocation

Mechanisms for handling congestion i.e. congestion control may be divided into congestion prevention, congestion avoidance and congestion recovery. Congestion prevention guides against congestion at all times while congestion avoidance disallows the possibility of the occurrence of congestion, and congestion recovery tries to restore an operating state when demand has already exceeded capacity. One of the congestion avoidance mechanisms developed is the Random Early Detection (RED) gateway for congestion avoidance with somewhat different method for detecting congestion.

To avoid congestion, resources are pre-allocated a in the case of Asynchronous Transfer Mode Network (ATM) of are allocated on-demand whether they are sufficient for the negotiated traffic or not as in he case of TCP/UDP traffics. In some Cases congestions are usually controlled if (and when) they occur. Some of the underlying service models include best-effort and multiple qualities of service. Figures 1 and 2 show the transmissions of packets from source to destination through single path and multiple paths respectively. In this research effort, we will consider single path model.

While the principle behind RED gateways are fairly general, and RED gateways can be useful in controlling the average queue size even in a network where the transport layer protocol cannot be trusted to be cooperative, RED gateways are intended for a network where the transport protocol

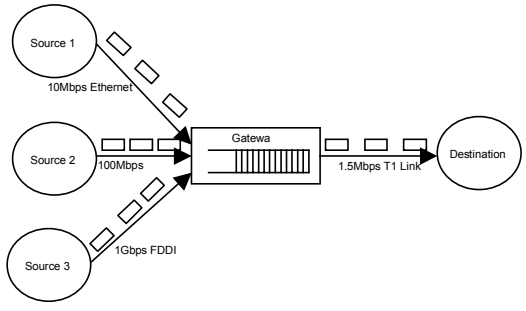

Figure 1: Packets transmission through single path

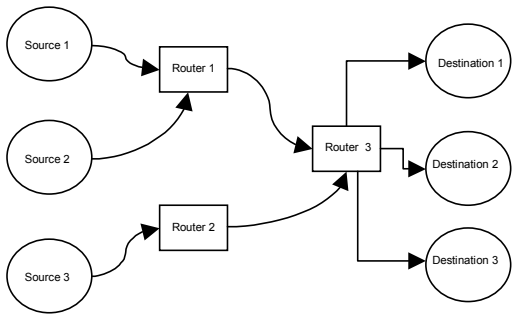

Figure 2: Packets transmission through multiple paths

responds to congestion indications from the network. According to Jacobson (1998), the Internet is an arbitrary mesh-connected network. The number of users that places demand on the network is not limited by any explicit mechanism; no reservation of resources occurs and transport-layer set-ups are not disallowed due to lack of resources. A path from a source to destination may have multiple hops, through several gateways and links as shown in Figure 2. The buffers for storing information flowing through Internet gateways are finite. The nature of the Internet protocol is to drop the packets when these buffers overflow.

Gateway congestion arises when the demand for one or more of the resources of the gateway exceeds the capacity of that resource. The resources include transmission links, processors and space used for buffering. Operationally, uncongested gateways operate with little queuing on the average, where the queue is the waiting line for a particular resource of the gateway. According to Kleinrock (1979), one commonly used quantitative definition when a resource is congested is when the operating point is greater than the point at which resource power is defined as the ratio of the throughput to delay.

Gateway Congestion Control Policies

There are several schemes for dealing with congestion at gateway. They differ in whether they use a control message and indeed, whether thy view control of the end-systems as necessary, but none of them in itself lowers the demand of users and consequent load on the network. Some of these are:

Source Quench- the method of gateway congestion control currently used in the Internet is the Source Quench message of the RFC-792. Internet Control Message Protocol (ICMP). When a gateway responds to congestion by dropping datagram, it may send an ICMP Source Quench message to the source of the dropped datagram. A significant drawback of this policy is that its details are discretionary, or alternatively, that the policy is really a family of varied policy.

Random Drop- this policy intends to give feedback to users whose traffic congests on the gateway by dropping packets on a statistical basis. The key to this policy is the hypothesis that a packet randomly selected from all incoming traffic will belong to a particular user with a probability proportional to the average rate of transmission of that user. A key problem of this policy is that dropping a randomly selected packet results in users, which generates much traffic having a greater number of packets dropped compared with those generating little traffic. This policy can be Random Drop for Congestion Recovery or Random Drop for Congestion Avoidance (Jacobson, 1998).

Congestion Indication- This is otherwise called the DEC Bit policy. It was developed at the Digital equipment Corporation by Jain, Ramakrishnan, and Chiu (1987), originally for the Digital Network Architecture (DNA). Like Source Quench, it uses explicit communications from the congested gateway to the user. However, it uses the lowest possible network resources for indicating congestion. Unlike the Source Quench, the information is communicated in a single bit, the Congestion Experienced (CE). This policy attempts to avoid congestion by setting the bit whenever the average queue length over the previous

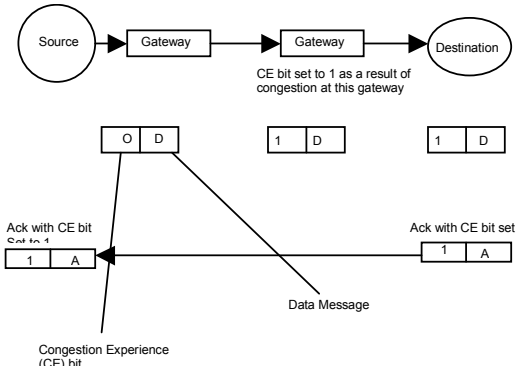

Figure 3: Network with Congestion Experience (CE) bit set to Indicate Congestion

queue regeneration cycle plus part of the current cycle is one or more, as shown in Figure 3. This policy can be Selective Feedback Congestion Indication. It works by keeping account of the number of packet sent y different users since the beginning of the queue averaging interval. This is similar to monitoring their throughputs. Based on the total throughput, a fair share for each user is determined and the congestion bit is set a described in Ramakrishnan, Jain, and Chiu (1987).

Fair Queuing- this is the policy of maintaining separate gateways output queues for individual end systems by source-destination pair. On congestions, packets are dropped from the longest queue. This policy leads to equal allocations of resources to end source-destination pair. This policy can be Bit-Round Fair Queuing or Stochastic Fairness Queuing (Demers, Keshav, & Shenker, 1989; McKenny, 1990).

Random Early Detection (RED)

Random early Detection is one of the active queue management control mechanism deployed at gateways. The RED gateway detects incipient congestion by computing the average queue size (Jacobson, 1998). The gateway could notify connections of congestions either by dropping packets arriving at the gateway or by setting a bit in packet headers. When the average queue size exceeds a preset threshold, the gateway drops or marks each arriving packet with a certain probability, where the exact probability is a function of the average queue size. RED gateways keep the average queue size low while allowing occasional burst of packets in the queue. Figure 4 show a network that uses RED gateway with a number of source and destination host while Table 1 shows RED gateway parameters and their meanings.

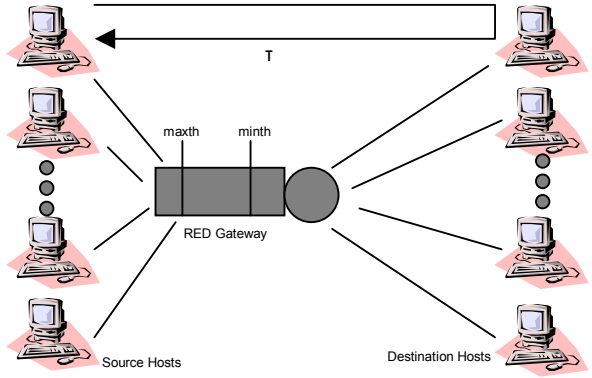

Fig.4: A network with RED gateway

Table 1: RED gateway parameters and their meanings

Parameter	Meanings
Min_{th}	Minimum threshold
Max_{th}	Maximum threshold
q_w	Weight factor for averaging
Max_p	Maximum packet marking probability
W(k)	Window size at slot k
N(k)	The number of TCP connections at slot k
τ	Propagation delay of TCP connections

The RED congestion control mechanism monitors the average queue size for each output queue, and using randomization, chooses connections to notify of the congestion. Transient congestion is accommodated by a temporary increase in the queue. Longer-lived congestion is reflected by an increase in the computed average queue size and result

In randomized feedback to some of the connections to decrease their windows. The probability that a connection is notified of congestion is proportional to that connection's share of the throughput through the gateway.

Related Researches

In Braden et al (1998), leading researches in the networking community have proposed implementation of RED in IP routers for Active Queue Management (AQM). Tuning of RED parameters has been a problem because of some difficulties presented in Christiansen, Jeffey, Ott, and Smith (2000) and in May, Bonald and Bolot (2000). In Firoiu and Borden (2000), the authors investigated the issue of recommendations of RED parameters and gave thumbs rules and guidelines for choosing them. Numerous RED variants (Clark & Fang, 1998; Feng, Kandlur, Saha, & Shink, 1999; Lin & Morris, 1997) have also been proposed, perhaps motivated by the difficulty in understanding the dynamics of RED with different parameter settings to the 'in' and 'out' packets of flows arriving at a router. Details of Diff Serv can be found in Blake et al. (1998). Closely related works to our proposed model are discussed in Lin and Morris (1997) and Ott, Lakshman, and Wong (1999). Lin and Morris (1997) used the concept of fair RED (FRED) which needs however to keep some per-active flow state. Its performance is ananlysed by simulating permanent TCP connections. Stabilized RED (SRED) was used in Ott, Lakshman, and Wang (1999), requiring per-flow state information too. But unlike RED, the drop probabilities depend only on the instantaneous buffer occupation and the estimated number of active flows. Fairness according to the transfer size is not considered there. May, Bonald, and Bolot (2000) analyses biasness with respect to bursty traffic. Both smooth as well as bursty traffic are modeled as Poisson processes, but in the smooth case each arrival corresponds to a single packet, whereas in the bursty traffic case each arrival brings a batch of packets.

In this research effort, we altered the original RED design guidelines that unconditionally allows transient congestion. This unconditional allowance of transient congestion is shown to be harmful when the queue is near full, because it causes buffer overflow at the gateway. Buffer overflow at the gateway leads to global synchronization and oscillation of traffic load on the network. To effectively prevent buffer overflow at the gateway, our proposed DRED model detects a transient congestion in a timely manner and take appropriate actions to quench it when the queue is near full. This is expected to solve the problem of the original RED where the departure of a packet does not cause any change to the average queue size.

System Model

The explosive growth of the Internet makes it essential to devise and deploy effective congestion control at the transport layer. The current Internet architecture is featured by the end-to-end TCP congestion control, in which congestion control is accomplished solely by end-hosts. The performance of end-to-end congestion control is expected to be greatly improved with the deployment of advanced gateway congestion-control mechanisms. The First In First Out queueing with the drop-tail policy is widely employed in the current Internet gateways, scheduling packets in a FIFO manner and discarding those packets arriving when the gateway buffer is full. The effects of global synchronization are reported to have been found in both one-way and two-way TCP traffic, and thus lower aggregate throughput.

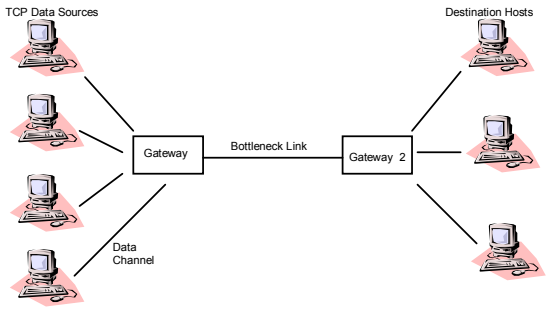

Fig. 5: A network model for implementing DRED algorithm

In this paper, we modeled the DRED architecture as shown in Figure 5. The network model consists of TCP hosts (sources and destinations), Gateways (G1 and G2), data channels from TCP source to the Gateway (G1) and from gateway (G2) to the destination hosts and the bottleneck channel between the two gate-

ways. The model components are described below:

- TCP Hosts: The TCP host consists of the data source and data destination for the network. It comprises of long-lived bulk-data transmission and interactive data transmission or short-lived bulk-data transmission. If the transient congestion is caused by long-lived bulk-data transmission, the transient congestion is very easy to become persistent in the near future if the DRED gateway accommodates the transient congestion without any negative feedback to traffic sources. However if the transient congestion is caused by short-lived transmissions or interactive-data transmissions, it may not be harmful when the queue is not near full because such a flow will terminate or will soon become idle.
- Data Channels: Data channel are the communication link between two or more hosts in a network. Each channel has its own capacity (i.e. amount of data that can pass through it at a time, measured in bit per second (bps) and propagation delay in second.
- The Gateway: The gateway is a device that connects networks using different communications protocols so that information can be passed from one point to the other. A gateway both transfers information and converts it to a form compatible with the protocols used by the receiving network. The RED model is design to be implemented on the gateway and so the enhancement, because congestion is easier determined by the gateway.

Model Analysis

There are two functions of a congestion-avoidance mechanism at a RED gateway: one is to detect incipient congestion, and the other is to decide which connections to be notified of congestion. In this research work we focus on how to detect incipient congestion by modifying the original RED design guideline of unconditionally allowing transient congestion (i.e., no negative feedback for transient congestion) which is harmful when the queue is near full.

The queue weight $\boldsymbol{w_q}$ is used to control the rate at which the estimated average queue size reacts to the changing traffic load at gateways. In the original RED, w_q is preset and remains unchanged after its deployment. By contrast, in the proposed approach, we monitor the dynamics of the actual queue and dynamically change the value of w_q according to the change of actual queue size. By reconfiguring w_q, we detect transient congestion in a timely manner when the queue is near full and take actions to quench it, thus avoiding buffer overflow.

In accordance with the reconfiguration of w_q, the framework of the RED algorithm is modified. The congestion-avoidance phase is extended and divided into a number of sub-phases. In contrast to the original RED in which the maximum drop probability is fixed, the refined RED; i.e. DRED, dynamically adjusts the value of maximum probability $\boldsymbol{max_p}$ depending on which sub-phase the current average queue length belong to. Since max_p directly impacts the aggressiveness of the early detection mechanism and average queue buildup indicates incipient congestion, we increase max_p as the average queue length increases from a lower sub-phase to a higher sub-phase. The illustration of the RED is shown in Figure 6.

Fig. 6: Protocol of the RED model

The RED and DRED Models

The RED model is to be deployed at a gateway for congestion avoidance and randomly drops packets before the gateway buffer is completely exhausted. RED aims to maintain high through-

put and low delay by controlling the average queue size, and avoid global synchronization and a bias against bursty traffic.

In the design of RED model, two preset thresholds; minimum threshold (min_{th}) and maximum threshold (**max_{th}**) are used to detect incipient congestion and control the average queue size. According to the estimated average queue length (avg **)**, a gateway operates in one of three different working states. When the average queue length is less than the minimum threshold (m_{th}), the gateway is in the ***green*** state. All incoming packets are processed and forwarded properly, and no packet is dropped. When average queue length is between the minimum and maximum thresholds, the gateway is in the ***yellow*** state. Arriving packets are randomly dropped with a probability that is a function of the average queue length. When the average queue length is greater than the maximum threshold, the gateway is in ***red*** state in which every arriving packet is discarded. The behaviors of RED in ***green*** and ***red*** states are the same as those of drop-tail. ***Yellow*** is the key state in RED where the congestion-avoidance mechanism is implemented.

The estimation of the average queue size and the calculation of drop probability are two key components of the RED algorithm. The success of RED depends on how to estimate the average queue size and set the drop probability. The filter used to compute the average queue size is an Exponentially-Weighted Moving Average (EWMA) given by:

$$avg \leftarrow (1 - w_q)avg + w_q q \quad (1)$$

Where w_q, the queue weight is a constant parameter preset by RED that determines the sensitivity of RED to the fluctuation of actual queue size, and q is the actual queue size.

In our DRED model, we introduce performance enhancements to the design of the original RED model in order to detect the initial congestion stage early. We therefore enlarge w_q to increase the responsiveness of RED to bursty traffic. However, a faster increase of w_q could result in overreaction to short-lived bursty traffic even if its burstiness is within the bound of the remaining free buffer space, biasing against short-lived bursty traffic. The surplus of actual queue size over average queue size given by Equation 2, which reflects the burstiness of the incoming traffic. Based on the surplus, the gateway can gain a useful hint about the incoming traffic. A large surplus means bursty incoming traffic. The continuous growth of the surplus indicates that the incoming bursty traffic is beyond the gateway's buffer capacity and buffer overflow is imminent.

$$surplus = q - avg \quad (2)$$

*Where **q** is the actual queue size and **avg** is the average queue size.*

If the surplus is low, the incoming traffic is less bursty. The transient congestion caused by small or short-lived bursty traffic should be accommodated since it does not cause buffer overflow.

We have two enhancements to improve the scalability of the RED algorithm. The first enhancement is to dynamically adjust the value of w_q with the change of the surplus of actual queue size over average queue size. The burstiness that the gateway can accommodate is determined by the actual buffer size. The larger the actual buffer size, the burstier the traffic to be accommodated. The surplus should be measured in the context of buffer size. The metric used here is the ratio of surplus to buffer size, which is given by Equation 3.

$$R = \frac{surplus}{buf\_s} \quad (.3)$$

Where buf_s is the gateway actual buffer size and R is the Ratio.

The Ratio R is a measure of burstiness and aggressiveness of the in coming traffic. Depending on the variations of this Ratio, w_q should be dynamically set.

The second enhancement is how to calculate the average queue size when a packet leaves the gateway. The original RED only estimates the average queue size upon each packet arrival, so, when no packet arrives, the dequeue operation is not captured. To correct this, we first compare the actual queue size with the average queue size. If the actual queue size is smaller than the average queue size due to the speed disparity caused by transient quiescence or large reduction of the incoming traffic between packet departures and arrivals, we use a larger w_q to calculate the average queue size in order to reflect the rapid dequeuing. With decrease of the average queue size, the drop probability is also reduced. When the next-round traffic arrives, fewer packets are dropped.

Dynamic Adjustment of Queue Weight

In the original RED, the queue weight w_q that is used to control the rate at which the gateway reacts to the congestion in the network is preset to a low-pass filter. As earlier mentioned, the original RED cannot react to highly bursty traffic fast enough to prevent buffer overflow. On the other hand, if wq is set too high, the RED algorithm will react too quickly to short-lived bursty traffic, causing bandwidth under-utilization and biasing against bursty traffic.

Based on the first enhancement in the previous section we dynamically adjust the value of w_q upon each packet arrival. In the DRED model, the dynamics of the actual queue size is monitored as shown in Equation 4. We introduce a new threshold called the ***warning line***, when measuring the actual queue size upon arrival of a packet. It divides the setting of w_q into two phases. If the actual queue size is below the warning line, w_q is set exactly the same as the original RED model value. However, as soon as the actual queue size is on or beyond the warning line, w_q is set as follows: *the higher the ratio of surplus to buffer size, the larger the queue weight.*

$$nw_q = \begin{cases} oldw_q, & R \in [0,0.1] \\ oldw_q \times 4, & R \in [0.1,0.2] \\ oldw_q \times 8, & R \in [0.2,0.3] \\ oldw_q \times 12, & R \in [0.3,0.4] \\ oldw_q \times 16, & R \in [0.4,0.5] \\ oldw_q \times 20, & R \in [0.5,1] \end{cases} \quad (4)$$

Where R is the ratio of surplus to buffer size and warning line is set to half of the buffer size. nw_q is the new queue weigh and $oldw_q$ is the old queue weight.

Upon departure of a packet from the queue, w_q is set according to the second enhancement in Equation 4. If the actual queue size is smaller than the average queue size, w_q is set to 0.02, instead of 0.002, in order to be responsive to rapid dequeueing. A detailed algorithm for computing w_q is shown in Figure 7, and the meaning of the parameters used by the DRED model is as shown in Table 2. The key point is to detect the initial stage of congestion quickly when the queue is near full and the incoming traffic is highly bursty, but still absorb short-lived or interactive traffic as best as possible.

```
For each arriving packet P:
q++,
if (q>avg)
    diff = q – avg ;
else
 diff = 0;

ratio = diff / buf_s
R = int (10*ratio);

If (q < warn_line)
     nw_q = oldw_q
  else {
            switch (R) {
              case 0:  nw_q = oldw_q
              case 1:  nw_q = oldw_q *4
              case 2:  nw_q = oldw_q *8
              case 3:  nw_q = oldw_q *12
              case 4:  nw_q = oldw_q *16
              default:  nw_q = oldw_q *20
```

```
For each departing
 q--;
if (avg > q)
 nw_q = oldw_q*10;
 else
 nw_q = oldw_q;
```

Fig. 7: A detailed algorithm for computing w_q

Table 2: DRED parameters and their meanings

Parameter	Meanings or value
Old_q	0.002
Buf_s	buffer size
warn_line	half of buffer size
Q	actual queue size
Diff	surplus of actual queue size over average queue size
Ratio	ratio of surplus over buffer size
R	the integer part of (10*ratio)
nw_q	new queue weight

Resetting of Maximum Drop Probability

The performance of RED is very sensitive to the maximum drop probability max_p, which determines the aggressiveness of RED towards incoming packets when it is in **yellow state**. The need for varying max_p according to the change of average queue size is to lower queueing delay and avoid buffer over-flow. Since the build-up of average queue size indicates the imminence of persistent congestion, the RED gateway should be more aggressive when the average queue size increases.

The max_p in the original RED is fixed and remains at the fixed value regardless of the estimated average queue size. The packet-drop probability $\mathbf{p_b}$ is given by Equation 5. With the change of average queue size, $\mathbf{p_b}$ varies linearly from **0** to $\mathbf{max_p}$. However, the fixed maximum drop probability of the original RED does not work well for different traffic loads. In the DRED model, the setting of the maximum drop probability is adjusted accordingly. With the change of average queue size, max_p is dynamically switched to different settings. Within each sub-phase, the value of max_p is also fixed and $\mathbf{p_b}$ varies linearly from **0** to $\mathbf{max_p}$.

$$p_b = \frac{\max_p \left(avg - \min_{th}\right)}{\left(\max_{th} - \min_{th}\right)} \qquad (5)$$

System Simulation and Results Discussion

From the mathematical model developed, simulations were performed for the DRED model. The simulations were also performed for the original RED model using MATLAB 6.5 to compare their performance. The specific condition for the simulation is that the packets are generated randomly.

Simulation of the RED and DRED Architectures

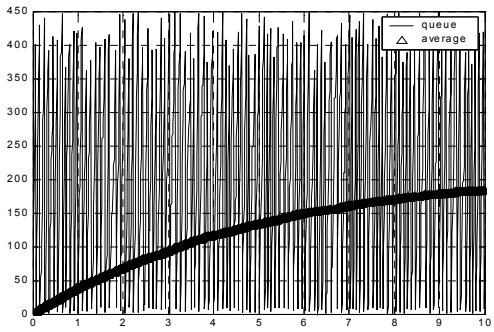

Figure 8: Simulation of RED gateway model

The RED model is simulated with queue weight $w_q = 0.002$, minimum threshold $min_{th} = 100$, and maximum threshold $max_{th} = 450$. The result of the simulation is as shown in Figure 8. The DRED model developed is simulated with w_q, min_{th} and max_{th} set as those of RED above and with additional parameters buffer size buf_size = 500, warnline = ½(buf_size). The packet is generated randomly with the function RAND in MATLAB. This function generate a random number X such that 0<X<1 and this is multiply by 10 or 100 to get a whole number that can be used to represent number of packets that arrive at the gateway.

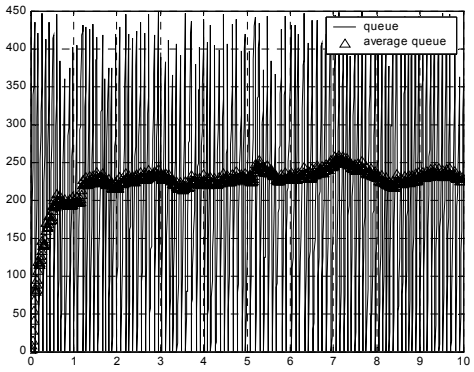

Figure 9: Simulation of DRED gateway model

The initial of average queue ***avg*** is zero and subsequent vales were calculated, the average queue ***avg*** and the current queue are then plotted against time as shown in Figure 9. Instead of continuous adjustment of $\boldsymbol{w_q}$, and $\boldsymbol{max_p}$ they are discretely set to different levels. At each level w_q and $\boldsymbol{max_p}$ are fixed to absorb the disturbance caused by short bursty traffic. However if the burstiness of the incoming traffic lasts long enough, the gateway should automatically push w_q and $\boldsymbol{max_p}$ to a more aggressive level in order to quench the burstiness of incoming traffic for avoidance of buffer overflow.

Performance Comparison

In this section, we presented the results of the simulations of the RED and DRED models as well as the discussions of the results obtained. We used two performance metrics i.e. the average queue and the probability of dropping packets.

The Average Queue

Figures 8 and 9 show the dynamics of average queue for both RED and DRED models respectively. As can be seen from the graph of DRED gateway (i.e. Figure 9), the average queue that determines the probability with which a packet is dropped responds early enough to the increased number of packets in the gateway (i.e. when the ratio of surplus [queue - avg] to buffer size in-

creases). Buffer overflow is effectively avoided in the DRED algorithm, and hence no global synchronization occurs and fluctuation of actual queue is greatly reduced.

However, from Figure 8, which shows the performance of the original RED model, the average queue ***avg*** increased slowly as the number of packets in the queue increases without taking into consideration the burstiness of the incoming traffic, which may lead to a situation whereby the entire buffer get filled thereby making RED gateway to degrade to droptail gateway and eventually leads to global synchronization and oscillation of the network.

The Probability of Dropping Packets

The probability with which a packet is dropped or marked for dropping is a function of the average queue at any point in time. Figures 10 and 11 shows the behaviours of RED and DRED gateway models respectively in terms of the probability of dropping packets as a function of average queue. As can be seen from the two graphs, the probability $\mathbf{P_b}$ remains at zero value until the average queue exceeds the minimum threshold (i.e. until the gateway enters the congestion avoidance phase). From the graph that represents the RED gateway model (i.e. Figure 10), the maximum probability that is obtainable from the simulation is about 0.005 which is not aggressive enough to handle bursty incoming traffic, which may lead to buffer overflow if the incoming traffic is from long lived bulk data transmission hence it eventually degrade the gateway to droptail. However, from the graph of DRED model gateway (i.e. Figure 11), the maximum drop probability is around 0.09. This shows the improvement of the DRED over the RED because probability is high enough to notify some of the sources picked randomly to reduce their rate of transmission so as to avoid buffer overflow.

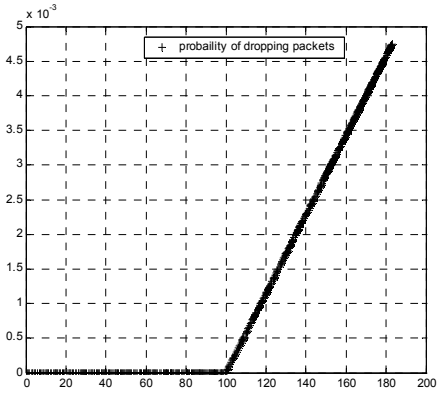

Fig. 10: Probability of dropping packets as a function of average queue for RED gateway model

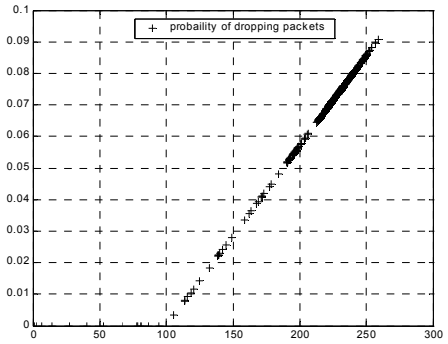

Fig. 11: Probability of dropping packets as a function of average queue for DRED gateway model

Figures 12 and 13 show the probability of dropping packets as a function of time for RED and DRED gateway models respectively. From the graph of RED model (i.e. Figure 12), the probability of dropping packets increases slowly and this may lead to buffer overflow if the incoming traffic is bursty. From Figure 13, it is shown that the probability of dropping packets for DRED model is responsive enough to the traffic at the gateway and as a result there will not be buffer overflow at the gateway.

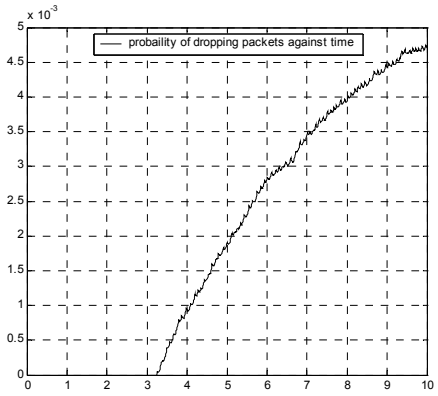

Figure 12: Probability of dropping packets as a function of time for dRED gateway model

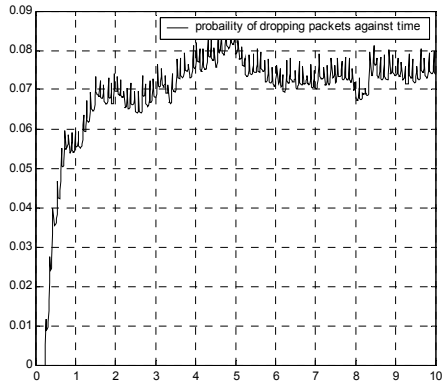

Figure 13: Probability of dropping packets as a function of time for DRED gateway model

Conclusion and Future Work

In this paper, we have examined the behaviour of Dynamic Random Early Detection (DRED) gateway and how it can be used to avoid congestion at the Internet gateways. This is because the most effective and appropriate point to detect congestion is at the gateway itself. The gateway can reliably distinguish between propagation delay and persistent queueing delay. Transient congestion is shown to be harmful when the queue of the gateway is near full. To prevent buffer overflow, a gateway must therefore be responsive to transient congestion when the incoming traffic is highly bursty and the free buffer space falls below our newly introduced parameter, warning line. A simple and efficient method to measure the burstiness of incoming traffic has been developed and tested. Based on the aforementioned, the estimation of average queue size *avg is* dynamically adjusted. The results of the simulation shows that the DRED model responds early enough to the increased number of packets at the gateway, hence global synchronization is avoided and fluctuations of actual queue size is greatly reduced.

Also, the maximum drop probability of the packets against the average queue size and time respectively shows improved performance over the original RED model. These results show that the DRED model responds early enough to avoid buffer overflow at the gateway when the queue is near full.

In future, we hope to investigate how routing decisions can be incorporated into the DRED model and we will also consider different output links (or multiple paths) in a situation where we have several gateways as in the case of the Internet architecture.

References

Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z. & Weiss, W. (1998). An architecture for differentiated services. Internet RFC-2475. Available at http://rfc.sunsite.dk/rfc/rfc2475.html

Braden, B., Clark, D., Crowcroft, B., Davie, S., Deering, S., Estrin, D., Floyd, S., Jacobson, V., Minshal, G., Patridghe, C., Peterson, L., Ramakrishnan, K., Shenka, S., Wroclawski, J., & Zhang, L. (1998). Recommendations on queue management and congestion avoidance in the Internet. RFC 2309.

Christiansen, M., Jeffey, K., Ott, D., & Smith, F. D. (2000). Tuning RED for web traffic. In *Proceedings of ACM/SIGCOMM*.

Clark, D. & Fang, W. (1998). Explicit allocation of best effort packet delivery service. *IEEE/ACM Transaction on Networking, 6*, 362-372.

Demers, A., Keshav, S., & Shenker, S. (1989). Analysis and simulation of a fair queueing algorithm. *Proceedings of the SIGCOMM Symposium on Communications, Architectures, and Protocols, 19* (4), 1-12.

Feng, W., Kandlur, D., Saha, D., & Shink, K. (1999). A self-configuring RED gateway. In *Proceedings of IEEE/INFOCOM*.

Firoiu, V. & Borden, M. (2000). A study of active queue management for congestion control. In *Proceedings of IEEE/INFOCOM*.

Floyds, S. & Jacobson, V. (1993). Random early detection gateways for congestion avoidance. *ACM/IEEE Transactions on Networking, 1*, 397-413.

Jacobson, V. (1998). Congestion avoidance and control. In *Proceedings of ACM SIGCOMM*, pp. 314-329.

Jain, R., Ramakrishnan, K., & Chiu, D. (1987). Congestion avoidance in computer networks with a connectionless network layer. Technical Report DEC-TR-506, Digital Equipment Corporation.

Kleinrock, L. (1979). Power and deterministic rules of thumb for probabilistic problems. *Computer Communications*. Proceedings of ICC.

Lin, D. & Morris, R. (1997). Dynamics of random early detection. In *Proceedings of ACM/SIGCOMM*, pp. 127-137.

May, M., Bonald, T., & Bolot, J. C. (2000). Analytic evaluation of RED performance. In *Proceedings of IEEE/INFOCOM*.

McKenny, P. (1990). Stochastic fairness queueing. Proceedings of *IEEE Infocom*, pp.733-740. Piscataway, NJ: IEEE Press.

Ott, T., Lakshman, T., & Wong, L. (1999). SRED: Stabilized RED. *Proceedings of Infocomm, 3*, 1346-1355.

Ramakrishnan, K., Jain, D., & Chiu, D. (1987). A selective binary feedback scheme for general topologies. Technical Report DEC-TR-510, Digital Equipment Corporation.

Biographies

A.A. Akintola obtained his Bachelors and Masters degree in Computer Engineering and Computer Science from Obafemi Awolowo University, Ile-Ife, in 1994 and 2002, respectively. He is a registered computer engineer with Council for the Regulation of Engineering Practice in Nigeria (COREN) and also a member of Nigerian Society of Engineers (NSE). He is an author of many journal articles in Nigeria and abroad. His current research interests are in the areas of teletraffic engineering, mobile/nomadic computing, digital computer networks, hardware system studies, and computer modeling and simulation of streaming video. He is also into the areas of wireless IP networks, error control, and curriculum development. He has over 7 years of experience in teaching and research. He is currently a Ph.D. student and a lecturer at the Department of Computer Science and Engineering of the same university.

G.A. Aderounmu holds a research degree M.Sc./PhD in Computer Science from Obafemi Awolowo University, Ile-Ife, Nigeria (1997 and 2001, respectively). He is a member of the Nigerian Society of Engineers (NSE) and is also a registered computer engineer with Council for the Regulation of Engineering Practice in Nigeria (COREN). He is also a Member of Nigerian Computer Society (NCS) and Computer Professional Registration Council of Nigeria (CPN). He has over 12 years of experience in teaching and research. He is an author of many journal articles in Nigeria and abroad. His special interests include engineering education in Nigeria, curriculum development, and computer communication and network. He is a Visiting Research Fellow to the University of Zululand, Republic of South Africa. He is currently a Senior Lecturer and the Acting Head of the Department of Computer Science and Engineering of the same university.

L. A. Akanbi obtained his Bachelor of Science degree in Computer Engineering from Obafemi Awolowo University, Ile-Ife, in 2003. He is a student member of Nigerian Society of Engineers (NSE). His current research interests are in the areas of error control schemes in teletraffic engineering and wireless communications. He is also into mobile IP networks, error control, and congestion control issues. He is currently on National Youth Service.

M. O. Adigun holds a research degree PhD in Computer Science from Obafemi Awolowo University, Ile-Ife, Nigeria, which he obtained in 1989. Currently, he is a Professor and Head, Department of Computer Science, University of Zululand, Republic of South Africa. His research interest include Software Engineering, Mobile Computing, Modeling and Simulation, and Performance Analysis of Computer System