Scalable Configuration of RED Queue Parameters [1]

K. Chandrayana, B. Sikdar and S. Kalyanaraman
Department of ECSE, Rensselaer Polytechnic Institute
Troy, NY 12180 USA
email: {kartikc,bsikdar,shivkuma}@networks.ecse.rpi.edu

Abstract— **Active queue management policies and in particular Random Early Drop (RED) are being pushed as additional mechanisms in Internet routers to control congestion and keep network utilization high. RED's performance is highly dependent on the settings of its control parameters. However, no firm guidelines exist on configuring RED parameters and the current suggestions fail to provide the desired performance scalability. In this paper, we propose a mechanism to configure RED parameters based on evaluating the expected instantaneous length of a $M^x/M/1/K$ queue. We show that by setting the RED parameters min_{th} and max_{th} to lie on either side of this expected queue length, we can ensure that the queue is not underutilized and flows cut their rates before the onset of congestion. This setting also allows the operating point to perform satisfactorily over a wide range of flow counts thereby allowing for a higher degree of scalability. We also show that our proposed mechanism increases the queue goodput, reduces losses and timeouts and increases the fairness when compared to existing guidelines. Our proposals have been verified using extensive simulations.**

I. Introduction

Traditional buffering techniques in routers are classified as "passive" and include TailDrop queues which accept packets till the maximum queue length is reached and then drop all subsequent packets. In a recent RFC [1] by the IETF, "active" queue management schemes at the routers were recommended with *random early detection* or RED queues [4] as the suggested buffering scheme. RED queues probabilistically drop incoming packets even though the queue is not full causing the TCP connections sharing the queue to reduce their transmission rates thereby ensuring that the queue does not overflow.

Though RED is being pushed forward for deployment across the Internet, guidelines for configuring RED parameters remains an open issue. Recent studies have shown that careful tuning of RED parameters is required to extract the benefits promised by RED and for RED queues to yield performance superior to TailDrop queues [2], [7]. The dependence of the queue performance on the operating point also leads to the problem of scalability. The original guidelines for tuning RED queues was presented in [4]. More recent recommendations on setting RED parameters are reported in [5] and [10]. These guidelines for setting RED parameters are based on heuristics which fail to provide desired performance over a wide range of traffic scenarios.

In this paper we propose a mechanism to configure RED queues which can also be used to dynamically tune RED parameters based on the current load. Our mechanism is based on setting parameters based on the expected length of a $M^x/M/1/K$ queue without early detection, fed with an arrival process which closely approximates TCP traffic. Using this expected length to set RED parameters min_{th} and max_{th} ensures that the queue and the output link is not underutilized. Setting the min_{th} and max_{th} on the either side of the expected queue length not only addresses the problem of scalability but also prevents the flows from ramping up their rates, by operating in the RED's drop region. Since our derivations account for the incoming traffic on a per flow level rather that the aggregate traffic, our methodology can also be used to dynamically configure RED parameters. The proposed configuration mechanism performs better than existing guidelines in terms of the queue goodput, loss rates, number of timeouts and the fairness amongst flows.

The rest of the paper is organized as follows. In Section II we present the existing guidelines for configuring RED queues. We then present our scheme for configuring RED in Section III. Section IV presents the simulation results while Section V presents the concluding remarks.

II. Background and Related Work

RED probabilistically drops packets even before the queue is full based on a weighted average of the total queue length. By using a weighted average queue as the decision variable, RED tries to avoid over-reactions to bursts and instead reacts to long term trends. The reader is referred to [4] for the detailed RED algorithm. RED offers five control parameters, $qlen$, max_p, min_{th}, max_{th} and w_q to tune RED's dynamics according to requirements. However, the impact of the choice of values of individual parameter on the queue's performance is dependent on the the values of the other too. Thus a judicious choice of parameter values is complicated and it is clear that simple heuristics are not sufficient. We now describe the RED configuration schemes proposed in literature.

Rough guidelines for configuring RED were presented in the original RED paper by Floyd and Jacobson [4]. It was suggested that w_q should be set greater than or equal to 0.002 and $min_{th} - max_{th}$ should be sufficiently large to avoid global synchronization. Also, min_{th} should be set sufficiently large to avoid underutilization of the output link. A more recent set of guidelines are presented in [5] which recommends that max_{th} should be three times min_{th}, max_p should be set to 0.1 and w_q should be set to 0.002. The proposal notes that the optimal setting for

[1] This work was supported by NSF contract number ANI9806660

min_{th} depends on the tradeoff between low average delay and high link utilization and suggest setting it to 5. But various experiments have shown that these parameters setting can still fail, specially when the queue is hit with many flows [11], as would be the case in practice.

In [10], Morris provides a RED configuration mechanism based on the number of flows traversing the queue. He first notes that to avoid excess timeouts, an ideal router for TCP should buffer more than four packets per flow. If a TCP keeps *w* packets in flight, with a round trip time *r*, the average packets in flight can be calculated as:

$$pif(l,r) \quad = \quad (1 - O(l,r,w(l)))w(l) \qquad (1)$$

where, $O(l,r,w)$ is the fraction of time spent in timeout and $w(l)$ is the window size. Then given there are n active flows and using the average packets in flight, the $\max_{th}$ can be chosen as

$$max_{th} = \frac{3}{2} \cdot n \cdot pif(l,r). \qquad (2)$$

At this point the the choice of $\max_p$ is limited to $max_p = \frac{3}{4}l$ which keeps the queue somewhat less full. The details of the derivation are given in [10].

III. RED Parameter Configuration

Recent studies have shown that the buffering should be proportional to the number of active flows [11]. Further it has been shown that a per-flow buffering of 5-6 packets greatly reduces the timeouts by ensuring that losses are recovered through fast-recovery and fast-retransmit algorithms [11]. Thus the two parameters which determine the performance of RED are $\max_p$ and $\max_{th}$ since they control the amount of buffering available per flow and the rate at which packets are dropped. Note that there is a tradeoff between a conservative/aggressive marking probability and fairness. While a conservative marking probability can be unfair, an aggressive marking policy, though fairer, increases the link loss rate thereby decreasing the link utilization. However, if max_{th} is quantified properly, an RED queue can be prevented from behaving like a Tail-Drop thereby making the choice of max_p easier.

In this section we focus on determining an appropriate value of max_{th} as a function of the number of active flows. Our technique is based on estimating the expected queue length of the RED queue without early and forced drops given a number of flows and setting the RED parameters min_{th} and max_{th} to lie on either side of this expected length. We model RED as $M^x/M/1/K$ queue. The model implies we have "bulk arrivals" (in the form of bursts of packets from the competing TCP sources) of varying sizes to the RED buffer. The interarrival time distribution of the bulks is given by an exponential distribution following the assumptions of [7]. The distribution of packets inside the bulk is modeled by a general distribution $g(x)$. The bulk size depends on the stationary window size distribution of the TCP sources and can vary from 0 to m, where m is dependent on the loss rates and the effect of TCP window limitation and has been characterized in [3], [6], [8], [9]. The processing times of the packets in the router are assumed to be exponentially distributed with mean $1/\mu$ and the offered load to the queue is thus $\rho = E[g(x)]\lambda/\mu$. The buffer size K corresponds to the RED parameter $qlen$ and is the maximum number of packets that the queue can accommodate. We assume that if the load is sufficiently high, then at the steady state, the average queue length and the expectation of the instantaneous queue length will be approximately the same.

Let state i, $0 \leq i \leq K$, denote that there are i packets in the queue and $p(i)$ denote the probability that there are i packets in the system. Assuming that the service rate is greater than the arrival rate, i.e. $\rho < 1$ (the condition for ergodicity), we may write the steady state equations as

$$\lambda p(0) = \mu p(1) \qquad (3)$$

$$(\lambda+\mu)p(r) = \lambda \sum_{i=1}^{r} p(r-i)g(i) + \mu p(r+1), \forall\ 1 \leq r \leq m-1 \qquad (4)$$

$$(\lambda+\mu)p(r) = \lambda \sum_{i=1}^{m-1} p(r-m+i)g(m-i) + \mu p(r+1), \forall\ m \leq r \leq K-1 \qquad (5)$$

$$\mu p(K) = \sum_{i=0}^{m} p(k-m+i) \sum_{j=0}^{i} g(m-j) \qquad (6)$$

with the constraints $\sum_{i=0}^{m} g(i) = 1$ and $\sum_{i=0}^{K} p(i) = 1$.

We use the method of maximum entropy [13] to calculate the probabilities $p(i)$ and the expected queue length. Our system can be characterized using four constraints: 1) normalization ($\sum_{n=0}^{K} p(n) = 1$), 2) utilization factor $\rho = \bar{\lambda}/\mu$ with $\sum_{n=0}^{K} \xi_0(n)p(n) = \rho$ where $\xi_0(n) = \min[1, max(0,n)]$ and $\bar{\lambda}$ is the effective arrival rate, 3) the expected queue length $E[Q] = \sum_{n=0}^{K} np(n)$ and 4) the probability that the queue is full, $p(K)$ which can be written as $\sum_{n=0}^{K} \xi_1(n)p(n) = p(K)$ where $\xi_1(n) = \max[0, n-K+1]$. We can now use Lagrange's method of undetermined multipliers to obtain the set of values for $p(n)$ that maximizes the entropy function $H(p) = -\sum_{n=0}^{K} p(n) log p(n)$ subject to the above four constraints. We get

$$p(n) \quad = \quad p(0)x^{\xi_0(n)}y^n z^{\xi_1(n)}, \qquad (7)$$

where x=$e^{-\theta_1}$, y=$e^{-\theta_2}$, z=$e^{-\theta_3}$, and θ_1, θ_2 and θ_3 are the Lagrangian multipliers corresponding to the constraints 2, 3 and 4. From Equation (7), we can obtain the following recursions

$$p(K) \quad = \quad zyp(K-1) \qquad (8)$$

$$p(n) = yp(n-1), \quad 1 \leq n \leq K-1 \tag{9}$$
$$p(1) = xyp(0) \tag{10}$$

The quantities x, y and z can be calculated by substituting Equation (7) into the steady state equations and we have

$$x \cdot y = \frac{\lambda}{\mu} \tag{11}$$
$$(\lambda + \mu) = \mu y + \lambda \sum_{i=1}^{r} y^{-i} g(i) \tag{12}$$
$$z = \sum_{i=0}^{m} y^{i-m-1} \sum_{j=0}^{i} g(m-j). \tag{13}$$

The above six equations define the solution to our queueing model. Note that the solutions are generic in nature and are valid for any arbitrary choice of the burst or window size distribution $g(x)$. Distributions to characterize $g(x)$ for TCP sources and to estimate λ from the number of active sources is presented later in the section.

A. Parameter Settings

The expected value of instantaneous queue will be in terms of the arrival rate and the expectation of distribution of the bulk size and is given by

$$f(\lambda, \bar{g}) = E[Q] = \sum_{i=1}^{K} ip(i) \tag{14}$$

where $\bar{g} = E[g(x)]$. To configure the RED parameters min_{th} and max_{th}, we propose placing them such that $E[Q]$ lies between them and is equidistant from from both min_{th} and max_{th}, i.e.,

$$\frac{max_{th} - min_{th}}{2} + min_{th} = E[Q] \tag{15}$$

This allows scalability of the parameter settings as the number of flows in the queue changes. For moderate increase in the number of flows, the expected queue length would still be less than max_{th} preventing the RED queue from behaving like a TailDrop while keeping the utilization high. On the other hand, if the number of active flows decreases, the queue length reduces, allowing the sources to increase their rates without causing congestion since the queue has been configured for a larger number of flows. Now, using the guideline $max_{th} = 3min_{th}$ [5] we can write min_{th} and max_{th} in terms of $E[Q]$ as

$$min_{th} = \frac{E[Q]}{2} \quad \text{and} \quad max_{th} = \frac{3E[Q]}{2} \tag{16}$$

The remaining three parameter settings are quite straightforward. The maximum queue length $qlen$ is determined by the physical system configuration and is the size of the queue. The maximum loss rate max_p is set according to the loss rate conditions prevailing in the network which should be around 2% under normal conditions and thus good guideline would be to set max_p between 0.05 and 0.1. The parameter w_q determines how fast the weighted average responds to the instantaneous queue lengths. In [4] quantitative guidelines for w_q are presented, in terms of the size of the transient burst that the queue can accommodate without dropping any packets at all and in general should be less than 0.005 with 0.002 being the default.

An RED queue starts behaving as TailDrop when the weighted average queue reaches max_{th} and all packets experience a forced drop. A rough estimate of the input arrival rate at which the expected queue length exceeds max_{th} and thus the range of the number of flows over which the parameter setting is valid can then be obtained by solving Equation 14 with respect to the arrival rate. So, the worst case scenario for the RED (when it becomes Tail-Drop) can be written as

$$\lambda = F(max_{th}, \bar{g}) \tag{17}$$

where *F* is the inverse mapping of the function *f*.

B. Arrival Rate as a Function of the Number of Flows

We now address the issue of estimating the batch arrival rate λ as a function of the number of active flows. We assume that the reader is familiar with the basic mechanisms of TCP like slow-start, timeouts, fast-retransmit and recovery etc. and is referred to [14] for further details. For simplicity, we consider the case when all the sources have the same round trip time (RTT). The case for heterogeneous RTTs is similar in nature. TCP transmits its packets according to a window based flow control mechanism and transmits a window's worth of packets every RTT. Thus the batch arrival rate corresponding to each TCP source is $1/RTT$. However, when a TCP source encounters a loss and goes into a timeout, no packets are sent till the retransmission timer expires. If we denote the fraction of time a TCP source spends in timeout by γ, the effective batch arrival rate corresponding to each source is thus $(1-\gamma)/RTT$. Now, if there are N flows in the queue, the total batch arrival rate is given by

$$\lambda = \frac{1-\gamma}{RTT} N \tag{18}$$

We use the derivations of [12] to estimate the fraction of time a TCP source spends in the timeout phase. From [12], a TCP source experiencing a loss rate of p has an expected timeout duration $E[TO]$ given by

$$E[TO] = T_O \frac{1 + p + 2p^2 + 4p^3 + 8p^4 + 16p^5 + 32p^6}{1-p} \tag{19}$$

where T_O is the period of time a sender waits before retransmitting an unacknowledged packet. We denote the

probability that an arbitrary loss leads to a timeout by Q and the expected duration of a congestion avoidance phase (where packets are transmitted every RTT) is denoted by $E[CA]$. Q and $E[CA]$ can be expressed as

$$Q = \min\left(1, \frac{(1+(1-p)^3(1-(1-p)^{E[W]-3}))}{(1-(1-p)^{E[W]})(1-(1-p)^3)^{-1}}\right) \tag{20}$$

$$E[CA] = \begin{cases} \left(\frac{b}{2}E[W_u]+1\right)RTT & \text{if } E[W_u] < W_{max} \\ \left(\frac{b}{8}W_{max} + \frac{1-p}{pW_{max}} + 2\right)RTT & \text{otherwise} \end{cases} \tag{21}$$

where $E[W] = \min(W_{max}, E[W_u])$ is the expected value of the window size, $E[W_u] = \frac{2+b}{3b} + \sqrt{\frac{8(1-p)}{3bp} + \left(\frac{2+b}{3b}\right)^2}$ is the expected value of the unconstrained window size and W_{max} is the receiver's advertised maximum window size. The fraction of the time the TCP source spends in timeout is then

$$\gamma = \frac{QE[TO]}{E[CA] + QE[TO]} \tag{22}$$

C. Window Size Distribution

The window size distribution of TCP flows in ideal congestion avoidance (TCP flows without timeouts) has been investigated in [6] and similar results have also been obtained in [10]. Misra and Ott [8] extend the analysis of idealized TCP connections for the cases of variable packet loss rates for RED like queues. In [9] this model is extended to model the window size distribution for multiple persistent flows.

While the models described above are obtained using analytic derivations using a number of simplifying assumptions, the window size distribution of TCP flows based on empirical measurements is presented in [3]. Using measurements conducted on a bottleneck link connecting two corporate LANs it was shown that the TCP congestion window size can be approximated by a truncated normal distribution (i.e. with no negative values). The window distribution $g(x)$ used in the derivations of this section and of Section III can be characterized using any of the models mentioned above. In our simulations, we use the truncated Gaussian distribution of [3] as the distribution for $g(x)$.

IV. Simulation Results

To verify our results, we carried out extensive simulations using the network simulation *ns* [15]. The topology used for these simulations is shown in Figure 1 where the router deployed a RED queue. The TCP sources are of the TCP Reno family and do not employ delayed acknowledgments.

In the first set of experiments, we configure the RED queue for 48 TCP flows each with an RTT of $52ms$.

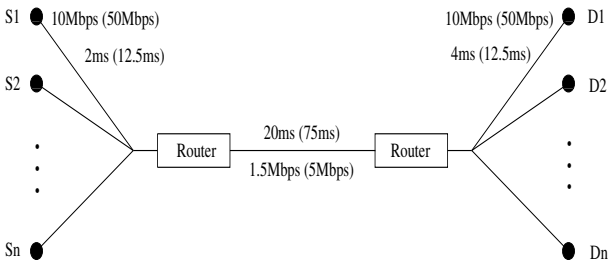


Fig. 1. Topology used in the experiments. The figure shows values for two sets of experiments with different link rates with the second set of values in the parentheses

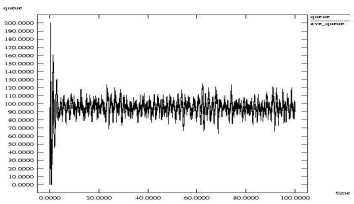
No. of Flows	Loss Rate		Timeouts		Fairness	
	Prop	Morr	Prop	Morr	Prop	Morr
32	5.17	6.35	366	840	0.10	0.19
48	5.72	6.05	642	1472	0.20	0.27
64	6.11	5.88	1113	1861	0.25	0.33

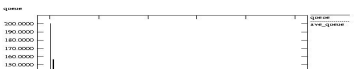
TABLE I

COMPARISON OF LOSS RATES (%), TIMEOUTS AND THE FAIRNESS OF OUR PROPOSED CONFIGURATION AND THAT PROPOSED IN [10].

Using the expressions of the previous section $E[Q]$ was calculated to be 79.54. $g(x)$ had a truncated Gaussian distribution with mean 4, variance 4 and the maximum burst size of 8. Following the guidelines of the previous section, the RED parameters were set to $min_{th} = 40$, $max_{th} = 120$, $max_p = 0.10$, $w_q = 0.002$ and the queue length $qlen = 200$. Figure 2(a) shows the simulation results for the instantaneous and average queue length of the RED queue set with these values and 48 sources. Note that our derivations predict the expected queue length quite closely and max_{th} is configured high enough to prevent forced drops. Figures 2(b) and 2(c) show the simulation results for the RED queue configured with the same values but with 32 and 64 flows. Note than even with a 33% change in the number of flows, our configuration keeps the average queue length less than the configured max_{th} showing the scalability of our parameter configuration. Figure 2 also shows the results for an RED queue configured with the guidelines in [10]. Following the guidelines of [10], $min_{th} = 20$ and $max_{th} = 60$ with the other parameters being the same. Note that this scheme fails to keep the queue from behaving as a Tail-Drop queue and the average queue stays at 60 most of the time, for all three cases. Also, note that the configuration is not scalable as is evident from the graphs for 32 and 64 sources in Figure 2. In Table I we compare the performance of the queue with our settings with those from [10] in terms of the goodput, the drop rates experienced by the flows, number of timeouts experienced by the flows and the fairness. We define fairness as the coefficient of variation of the throughputs of the various sources. Thus a lower value for the fairness index implies a fairer queue. Note that our configuration performs better than that of [10] in almost all cases.

Please note that the proposed value of $min_{th} = 5$ and $max_{th} = 20$ in [5] fail to be practical even for moderate


qlength
200.0000
190.0000
180.0000
170.0000
160.0000
150.0000
140.0000
130.0000
120.0000
110.0000
100.0000
90.0000
80.0000
70.0000
60.0000
50.0000
40.0000
30.0000
20.0000
10.0000
0.0000
0.0000
20.0000
40.0000
60.0000
80.0000
100.0000
qlength
R-VM_qlength
time


qlength
200.0000
190.0000
180.0000
170.0000
160.0000
150.0000
qlength
R-VM_qlength