Active Queue Management using Adaptive RED

Rahul Verma, Aravind Iyer and Abhay Karandikar

Abhay Karandikar is the corresponding author. He is with the Information Networks Laboratory, Department of Electrical Engineering, Indian Institute of Technology, Bombay, India, e-mail: karandi@ee.iitb.ac.in

Abstract

Random Early Detection (RED) [1] is an active queue management scheme which has been deployed extensively to reduce packet loss during congestion. Although RED can improve loss rates, its performance depends severely on the tuning of its operating parameters. The idea of adaptively varying RED parameters to suit the network conditions has been investigated in [2], where the maximum packet dropping probability max_p has been varied. This paper focuses on adaptively varying the queue weight w_q in conjunction with max_p to improve the performance. We propose two algorithms viz., $w_q{-}thresh$ and $w_q{-}ewma$ to adaptively vary w_q. The performance is measured in terms of the packet loss percentage, link utilization and stability of the instantaneous queue length. We demonstrate that varying w_q and max_p together results in an overall improvement in loss percentage and queue stability, while maintaining the same link utilization. We also show that max_p has a greater influence on loss percentage and queue stability as compared to w_q, and that varying w_q has a positive influence on link utilization.

Index Terms

Random Early Detection (RED), active queue management, next generation Internet.

I. INTRODUCTION

The problem of congestion control in TCP/IP networks has been studied extensively in the literature [3]. TCP provides an inherent mechanism for end-to-end congestion control. The essence of this mechanism is that a TCP source adjusts its window size based on an implicit feedback of a packet drop. Traditionally, a packet gets dropped whenever it encounters a full buffer (known as the Drop-Tail gateway). This however has been found to be inadequate owing to the following two reasons. First, gateways are designed with large buffers to accommodate transient congestion. However, queues that are occupied most of the time would be undesirable as this would increase the average delay in the network. Secondly, in spite of all the enhancements TCP has undergone, Drop-Tail gateways suffer from high packet loss rates. The reasons attributed to this high loss rate include a lack of early congestion notification to sources. This problem has been addressed by deploying active queue management strategies like RED in the network.

The basic philosophy behind RED is to *prevent* congestion rather than to *cure* it. In RED scheme, packets are dropped even before the buffer is full, in order to notify sources if congestion is building up. The sources can then reduce their window sizes (or rates), thereby preventing

further packet loss. The scheme is described in detail in the next section. However, RED has several shortcomings, including a high degree of sensitivity towards its operating parameters, unfairness to flows with different round-trip times, and the problem of global synchronization.

Several studies have been performed to address the above shortcomings [2], [4], [5], [6], [7], [8]. The idea of adaptively varying RED parameters has been proposed in [2]. In [2], one of the RED parameters max_p is adaptively varied. The objective is to reduce the oscillations in the queue length. We have demonstrated through simulations that varying max_p also has a positive effect on the packet loss percentage. However, as illustrated in the simulation results in this paper, varying max_p has an adverse effect on the link utilization. Floyd et al. in [8] discuss algorithmic modifications to the self-configuring RED algorithm [2] for tuning max_p adaptively. Their objective is to control the average queue length around a pre-decided target. However, it may be noted that controlling the average queue size has a limited impact on regulating the packet loss rate.

Previous papers have considered adaptive RED schemes in order to tune max_p. This paper addresses the issue of adaptively varying w_q. We demonstrate that adaptively varying w_q has a positive effect on link utilization. We show that varying w_q, by the techniques illustrated in the paper, along with varying max_p, in a manner similar to that of [2], brings about an improvement in terms of packet loss rates and queue stability without adversely affecting the link utilization. We, therefore, suggest that both max_p and w_q be varied adaptively.

The rest of the paper is organized as follows. Section II explains the problem of parameter sensitivity and discusses an algorithm to modify max_p adaptively. In Section III, we propose some algorithms for adaptively varying w_q. Section IV studies the relative performance of these algorithms. Finally, Section V concludes the paper.

II. BACKGROUND

A. RED scheme

The RED scheme was initially described and analyzed in [1]. The RED gateways detect incipient congestion by computing the average queue size. RED computes the average queue length q_{avg} using an exponentially weighted moving average:

$$q_{avg} = (1 - w_q)q_{avg} + w_q q_{instantaneous} \quad (1)$$

The packets are dropped based on the above q_{avg}. The averaging process smoothens out temporary traffic fluctuations and allows small bursts to pass through unharmed, dropping packets only during sustained overloads. RED maintains two queue thresholds: min_{th} and max_{th}. If q_{avg} exceeds max_{th}, all incoming packets are dropped, whereas if q_{avg} is less than min_{th} no packet is dropped. If q_{avg} lies between the two thresholds, packets are dropped with a probability p which increases linearly from 0 (at min_{th}) to max_p (at max_{th}). By dropping packets before the buffer is completely full, the RED gateway attempts to warn the TCP sources of incipient congestion.

B. Performance dependence on parameters

The various control parameters of RED are listed below (Table I). The performance of a

TABLE I

RED PARAMETERS

Parameter	Function
w_q	Weight for calculating average queue
max_p	Maximum dropping probability
min_{th}	Lower threshold below which no packet is dropped
max_{th}	Upper threshold above which all incoming packets are dropped

gateway employing RED depends significantly upon its parameters (see [2], [7]). In [9], certain guidelines to decide the values of these parameters have been given. However, experiments have shown that it is difficult to find appropriate values of parameters that will enable RED gateways to perform equally well under different congestion scenarios. In cases where the parameters are not suitable for the given network traffic load, the performance of the RED gateway can approach that of the traditional Drop-Tail gateway. One approach to solve this problem is to update the RED parameters dynamically in order that they are suitable for the given network conditions.

C. Self-Configuring RED

An algorithm to modify the parameter max_p based on the average queue length is described in [2]. The basic idea is to modulate the aggressiveness of RED scheme by examining the variations in average queue length. If the average queue length oscillates around min_{th}, then the algorithm reduces the value of max_p by a factor α (i.e. $max_p = max_p/\alpha$), in order to make RED less aggressive. Similarly, if the average oscillates around max_{th}, then max_p is increased by a factor β (i.e. $max_p = max_p * \beta$). This makes RED more aggressive. The outline of the algorithm is given in Figure 1. In the rest of the paper, we will call this the max_p algorithm. The algorithm performs well by reducing the oscillations in the instantaneous queue length under different traffic conditions.

On the nth q_{avg} Update

- Calculate average queue size as
  $q_{avg_n} \leftarrow (1 - w_q)q_{avg_{n-1}} + w_q.q_{instantaneous}$
- If ($min_{th} < q_{avg_n} < max_{th}$)
  status = Between
- If ($q_{avg_n} < min_{th}$ && status!=Below)
  status = Below
  $max_p = max_p/\alpha$
- If ($q_{avg_n} > max_{th}$ && status!=Above)
  status = Above
  $max_p = max_p * \beta$
- Rest of the RED algorithm remains the same.

Fig. 1. Self-Configuring RED algorithm (max_p)

III. Adaptive RED

The Self-Configuring RED algorithm improves the performance in terms of the instantaneous queue stability by reducing oscillations. However, (as shown in the simulations below) it results

in a lower link utilization[1] under conditions of congestion. Our simulation results show that adapting the parameter w_q has a positive effect on link utilization under conditions of heavy load. This motivates one to look at algorithms which vary w_q adaptively in conjunction with max_p, in order to get better performance in terms of both link utilization and stability. As shown by the simulations, varying both w_q and max_p results in an improvement in queue stability and loss percentage without bringing down the link utilization. Also, the performance is considerably better in terms of loss percentage and stability as compared to the default[2] scheme or varying only w_q. We would, however, like to point out that varying w_q alone does not bring any significant improvement in terms of loss percentage or stability.

We propose to vary the parameter w_q adaptively based on the variations of the average queue length. If the change in average queue length is significant, then at the next q_{avg} update, greater weight should be given to the instantaneous queue length and hence w_q should be increased. Similarly, if the change is negligible, w_q should be reduced and greater weight should be given to the previous value of the average queue size. In other words, w_q controls how fast the network conditions are learnt. In this paper, we suggest two algorithms which can be used to vary w_q. The first algorithm ($w_q - thresh$, Figure 2) switches w_q between two levels and the second algorithm ($w_q - ewma$, Figure 5) varies w_q through an exponentially weighted moving average. Both of these are explained in the following subsections.

A. $w_q - thresh$ algorithm

The $w_q - thresh$ algorithm changes the aggressiveness of RED scheme by changing the value of w_q based on the changes in the average queue length. The idea is to have two values for w_q, one (w_1) corresponding to network conditions where the queue length is varying rapidly and the other (w_2) corresponding to conditions of stable queue length. Depending on whether the variation in q_{avg}, viz., $|\Delta avg|$ exceeds a threshold ρ or not, the value of w_q is set to w_1 or w_2 respectively. w_1 is chosen to be greater than w_2 because more weight should be given to the instantaneous queue length when the queue size is varying rapidly. Also, w_1 and w_2 are varied close to 0.0020 which is the default value of w_q in the non-adaptive RED scheme (as suggested in [1]).

[1]The total number of packets transmitted *successfully* on a particular link (for a given time interval) is taken as a measure of the link utilization for that link.

[2]In the paper, the *default* scheme refers to the original RED scheme [1].

On the nth q_{avg} Update

- Calculate average queue size as

  $q_{avg_n} \leftarrow (1 - w_q)q_{avg_{n-1}} + w_q.q_{instantaneous}$
- If $|\Delta avg| > \rho$

  $w_q = w_1$

  else

  $w_q = w_2$

  where $\Delta avg = q_{avg_n} - q_{avg_{n-1}}$ and $w_1 > w_2$

  ρ is some threshold value
- Rest of the RED algorithm remains the same.

Fig. 2. Adaptive RED algorithm (w_q: Thresholding)

The performance of the $w_q - thresh$ algorithm has been studied for various values of w_1, w_2 and ρ. One significant observation that we would like to mention is that the performance of the algorithm is not affected significantly for ρ lying in the range [0.5, 2] (see Figure 3). For $\rho > 2$, the algorithm tends to be the same as the default scheme with w_1 as the queue weight, since $|\Delta avg|$ does not exceed ρ. For lower values of the parameter ρ the performance varies a lot for small variations in ρ (see Figure 4). In the simulations carried out in the paper, we have chosen a value of $\rho = 1$ and it has been found to work well under various network scenarios considered in the simulations.

The performance varies marginally with respect to the parameter pair $\{w_1, w_2\}$. The simulations were carried out with $\{w_1, w_2\}$ being equal to {0.0022, 0.0018}, {0.0021, 0.0017}, {0.0024, 0.0020} and {0.0020, 0.0018}. However, the pair {0.0022, 0.0018} gives the best performance in most of the scenarios simulated in this paper compared to the other values. The results in the paper correspond to $\{w_1 = 0.0022, w_2 = 0.0018\}$ and $\rho = 1$.

B. $w_q - ewma$ *algorithm*

The $w_q - ewma$ algorithm regulates the aggressiveness of RED scheme, by allowing the queue weight w_q to take a *continuous* set of values unlike the $w_q - thresh$ algorithm in which w_q takes only two discrete values. The scheme works by observing the change in q_{avg} and modifying w_q

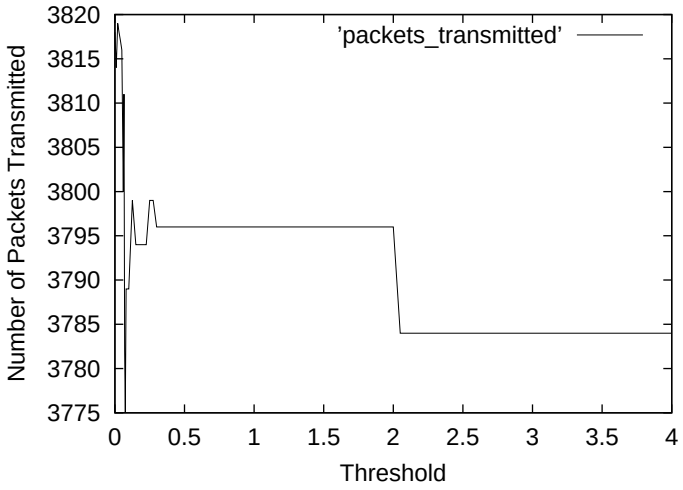

Fig. 3. Packets transmitted as a function of threshold ρ, ($w_1 = 0.0022$, $w_2 = 0.0018$) for a particular network scenario.

according to the equation

$$w_q \leftarrow k_1 w_q + k_2 |\Delta avg| \tag{2}$$

The constant $k_1 (< 1)$ is responsible for decreasing w_q whenever $|\Delta avg|$ is negligible. The constant k_2 changes w_q in proportion to $|\Delta avg|$ so that when the queue length is varying rapidly, w_q increases at a faster rate. The $w_q - ewma$ algorithm has been simulated for various values of the parameters k_1 and k_2. The algorithm gives a consistently good performance when k_1 is close to unity and k_2 is close to zero. Otherwise, the performance varies considerably and at times can become worse than the non-adaptive algorithm. Also, the parameters cannot be arbitrarily varied independent of each other without compromising on performance. From the simulations we determine that if k_2 is set close to $(1 - k_1)/10$ then the algorithm gives good performance for k_1 in $[0.999, 1.000]$. The rationale behind setting k_2 in terms of k_1 in such a fashion is the following.

The adaptation equation for changing w_q is given by Equation (2). As long as the change in the average queue length is observed to be negligible, ($|\Delta avg| \approx 0$), w_q keeps decreasing by a factor k_1. Whenever $|\Delta avg|$ is observed to be changing rapidly, w_q needs to be restored to a higher value. Therefore, k_2 needs to not only offset the decrease caused by k_1 (which is $(1 - k_1) * w_q$) but also provide the necessary increase. Hence, k_2 should be set to $(1 - k_1) * \frac{w_q}{|\Delta avg|} * \kappa$, where

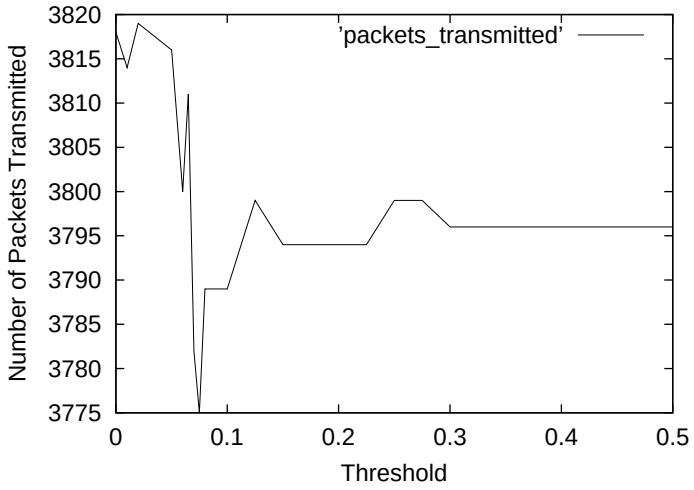

Fig. 4. Packets transmitted as a function of threshold ρ, (Magnified view: $\rho \epsilon$ [0,0.5], $w_1 = 0.0022$, $w_2 = 0.0018$) for a particular network scenario.

On the nth q_{avg} Update

- Calculate average queue size as
  $q_{avg_n} \leftarrow (1 - w_q)q_{avg_{n-1}} + w_q.q_{instantaneous}$
- Update queue weight w_q as
  $w_q \leftarrow k_1 w_q + k_2|\Delta avg|$
  where $\Delta avg = q_{avg_n} - q_{avg_{n-1}}$
- Rest of the RED algorithm remains the same.

Fig. 5. Adaptive RED algorithm (w_q: Weighted Average)

κ is a constant which controls the amount of increase in w_q. In addition, from simulations we find that the variations in the average queue length is about an order of magnitude higher than w_q. Therefore, k_2 can be set as $(1 - k_1) * \kappa/10$. We have taken κ to be 1.

In the simulations, when only $w_q - ewma$ is employed, we use the value $k_1 = 0.9996$ and hence $k_2 = 0.00004$, whereas when both the $w_q - ewma$ and the max_p algorithms are employed, then $k_1 = 0.9992$ and $k_2 = 0.00010$ are used. Here k_2 is set to 0.00010 instead of 0.00008, as dictated by the above reasoning. This is because the adaptation algorithm for max_p uses a value of α which is greater than β so that max_p decreases faster than it increases (see Section IV). So

a higher value of k_2 enables w_q to increase faster when congestion occurs and compensate for the slower response of max_p.

IV. Performance Evaluation

A. Experimental network

The simulations have been performed using the **ns v2.1b7** network simulator ([10]). The network topology shown in Figure 6 is used for simulating the various scenarios. It may be noted that the simulation topology used is similar to the topology used in other papers, for consistent comparison. The issue of multiple congested gateways has not been considered here. The value of N is varied for different scenarios. The default RED parameters are kept as $w_q = 0.0020$, $max_p = 0.1$, $max_{th} = 15$ and $min_{th} = 5$. The buffer size for the RED router is kept as 50 packets and the simulations are performed with equal sized packets. The results are obtained for the RED router and the bottleneck link shown in the figure.

We call a *harsh* scenario as one where the loss percentage (defined as number of packets lost / number of packets which entered the router) exceeds 20%. If the loss rate is around 1%, then the scenario is referred to as *mild*. In between (loss percentage $\approx$ 5-15%), the simulation scenario is called *moderate*.

In all of the simulations, the values of α and β for the max_p algorithm are taken as 2 and 1.5 respectively. The reasoning behind choosing these values is that max_p should be decreased more drastically compared to when it is increased. The packet loss rate is directly influenced by max_p. A large increase in max_p can lead to high loss rates and hence it should be increased cautiously. In [2], α and β are taken as 3 and 2 respectively. However, the values $\alpha = 2$ and $\beta = 1.5$ give a relatively better performance under harsh and moderate scenarios and these have been used in the simulations.

B. Harsh scenario

For the harsh scenario the total number of nodes has been taken as 80. The bottleneck bandwidth is 1Mbps while the other link bandwidths are 3Mbps. The various sources are switched on at random times between $t = 0$sec and $t = 7$sec. The sources stop transmitting at time $t = 30$sec.

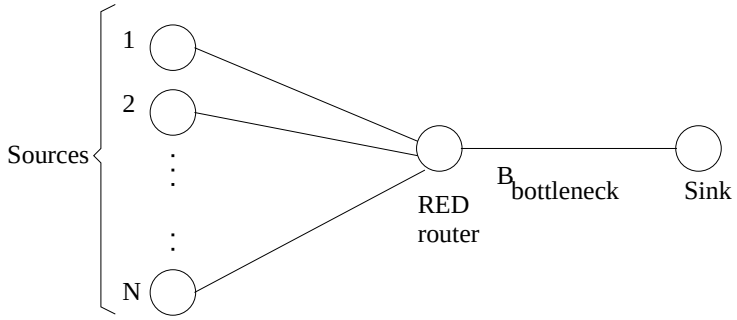

Fig. 6. Network Topology

TABLE II

PERFORMANCE COMPARISON (HARSH SCENARIO): $w_q{-}thresh$, $w_q{-}ewma$ AND max_p ALGORITHMS

Algorithm	Drop%	Packets	Stable
Default	30.39	3741.9	No
Only $w_q{-}thresh$	30.18	3750.1	No
Only $w_q{-}ewma$	30.23	3755.0	No
Only max_p	26.61	3724.9	Yes
$w_q{-}thresh$+max_p	26.19	3736.3	Yes
$w_q{-}ewma$+max_p	26.65	3730.0	Yes

From the simulation results (refer Table[3] II) it can be inferred that:

- Both the $w_q{-}thresh$+max_p and $w_q{-}ewma$+max_p algorithms are much better than the default scheme in terms of stability (see Figures 7, 8 and 9).
- Using only the max_p algorithm results in a drop in the number of packets transmitted successfully. Varying w_q in conjunction with max_p helps in achieving the benefits of varying max_p (lower loss percentage and more stability) and at the same time counters the adverse effect of max_p on link utilization. The overall improvement in performance is prominent in the moderate scenario.
- Varying only w_q results in an improvement in the number of packets transmitted. Even though the improvement seems marginal, in such a scenario where the link utilization is as such close to $97 - 98\%$, an improvement of only about half a percent is substantial. In

[3]By *packets*, we mean the number of *successfully* transmitted packets.

certain sample paths, the improvement was close to 2%. The importance of w_q in affecting the link utilization is further seen in the moderate scenario.

- Both the algorithms result in a substantial decrease ($\approx 4\%$) in the loss percentage compared to the default RED scheme.
- Also, the stability of the instantaneous queue length is better in the w_q-ewma+max_p algorithm than in the $w_q - thresh$+max_p. This can be attributed to the fact that w_q is intimately tied to the estimation of the average queue length and hence the regulation of the instantaneous queue length. So varying w_q continuously has a better stabilizing effect than switching it between two different values.

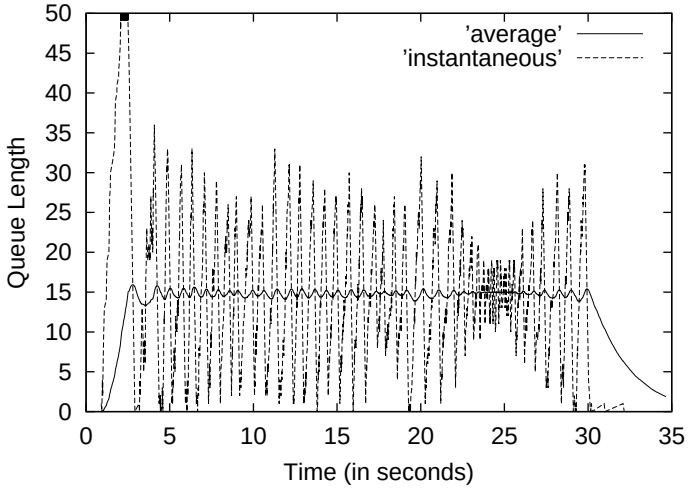

Fig. 7. Average and instantaneous queues with default RED (Harsh scenario)

C. Moderate scenario

Here the total number of nodes has been taken as 25. The sources start transmitting at random times between $t = 0$sec to $t = 5$sec. Their stop times are also random. The sources transmit till about $t = 20$sec. The following inferences can be drawn from the results (refer Table III):

- As far as the stability of the instantaneous queue is concerned, the problem is less severe in this scenario (Figure 10) compared to the harsh scenario. Still, the adaptive algorithms varying both max_p and w_q perform better (Figures 11 and 12) compared to the non-adaptive RED scheme.

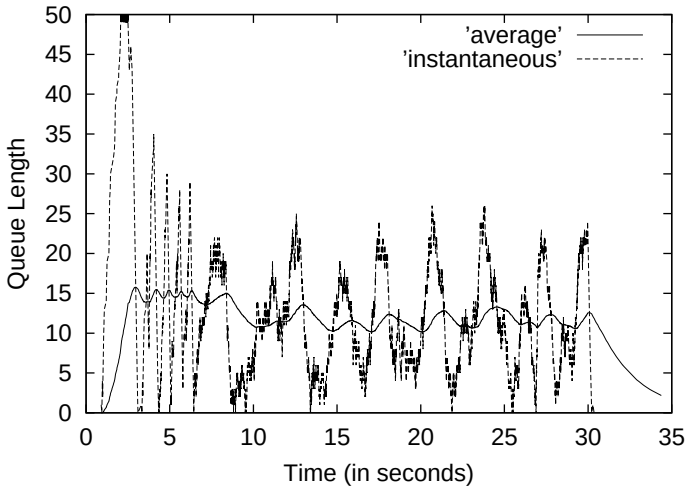

Fig. 8. Average and instantaneous queues with max_p+w_q-$thresh$ algorithm (Harsh scenario)

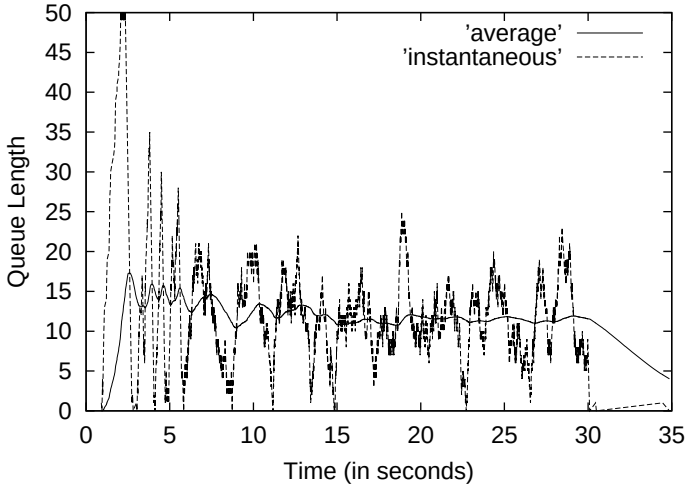

Fig. 9. Average and instantaneous queues with max_p+w_q-$ewma$ algorithm (Harsh scenario)

- The performance of the adaptive algorithms (varying both max_p and w_q) is better, compared to the default RED scheme, in terms of loss percentage and number of packets transmitted.
- Varying only max_p results in a drop in the number of packets transmitted. Varying w_q along with max_p results in a better link utilization compared to when only max_p is varied.
- Varying only w_q is not sufficient to bring down the loss percentage or make the queue stable.

TABLE III

PERFORMANCE COMPARISON (MODERATE SCENARIO): $w_q{-}thresh$, $w_q{-}ewma$ AND max_p ALGORITHMS

Algorithm	Drop%	Packets	Stable
Default	12.69	2369.3	No
Only $w_q{-}thresh$	12.44	2364.0	No
Only $w_q{-}ewma$	12.66	2362.9	No
Only max_p	11.84	2345.7	Yes
$w_q{-}thresh$+max_p	11.03	2372.1	Yes
$w_q{-}ewma$+max_p	11.94	2373.0	Yes

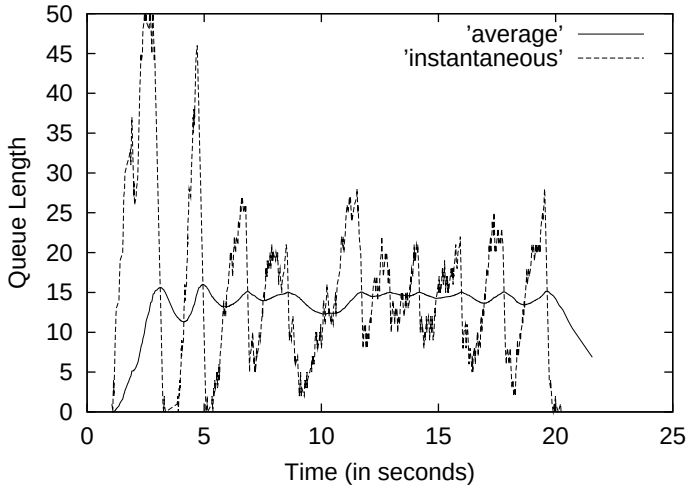

Fig. 10. Average and instantaneous queues with default RED (Moderate scenario)

D. Mild scenario

In a mild scenario, where the loss percentages are very low, the problem of stability does not arise. Also, using any of the adaptive algorithms does not help much in reducing the loss percentages further. However, there is an increase in the number of packets transmitted.

The results (refer Table IV) shown are for $N = 5$ and all link bandwidths being equal to 10Mbps. As can be seen from the results, varying only max_p brings about a good improvement in the number of packets transmitted and further varying w_q (using the $w_q{-}thresh$ algorithm) improves the performance only marginally. Thus, if a tradeoff has to be made between com-

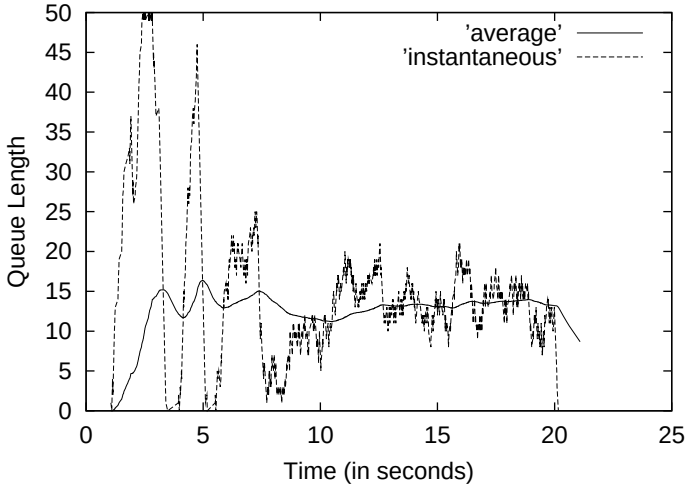

Fig. 11. Average and instantaneous queues with max_p+w_q-$thresh$ algorithm (Moderate scenario)

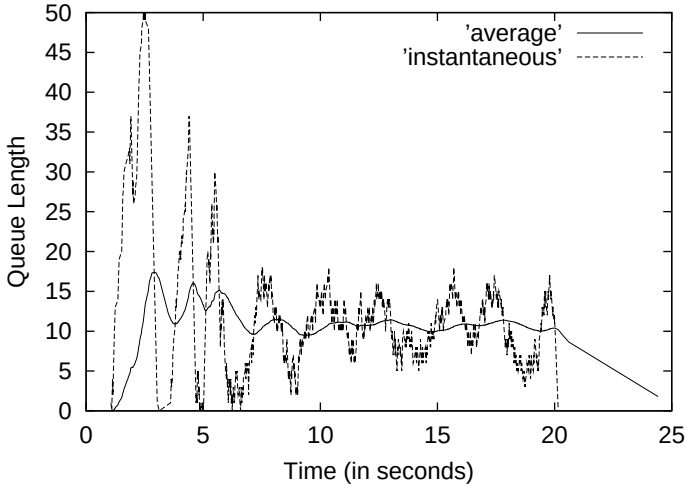

Fig. 12. Average and instantaneous queues with max_p+w_q-$ewma$ algorithm (Moderate scenario)

plexity and performance, the results suggest that only max_p should be varied in such a scenario. However, varying both w_q ($w_q-thresh$ algorithm) and max_p still gives a better performance compared to the other schemes. The w_q-ewma algorithm gives poorer performance in terms of the packets transmitted but there is marginal improvement in the loss percentage. This can be explained as follows. The manner in which w_q is varied controls how fast the network conditions are learnt. Since in a mild scenario, the network changes are smaller, the $w_q - thresh$ algorithm

is faster to respond than the $w_q - ewma$ algorithm, and hence performs better. This suggests that in scenarios with little or no congestion, w_q should be varied according to the $w_q - thresh$ algorithm. In case computation complexity is of great concern, then only max_p should be varied in such scenarios.

It was also observed that in mild scenarios, varying max_p in larger steps (e.g. $\{\alpha = 3, \beta = 2\}$) gives better results than varying it with $\{\alpha = 2, \beta = 1.5\}$. However, α and β should not be too large because then the algorithm becomes unstable.

TABLE IV

PERFORMANCE COMPARISON (MILD SCENARIO): $w_q{-}thresh$, $w_q{-}ewma$ AND max_p ALGORITHMS

Algorithm	Drop%	Packets
Default	0.80	24715.6
Only $w_q{-}thresh$	0.82	24713.2
Only $w_q{-}ewma$	0.77	24387.7
Only max_p	0.73	24774.5
$w_q{-}thresh$+max_p	0.73	24789.7
$w_q{-}ewma$+max_p	0.61	24531.2

E. Summary of results

Based on the above observations, we can make the following conclusions on the effect of max_p and w_q on the performance of RED:

1) max_p has a stronger influence on the loss percentage and stability of RED as compared to w_q. The strong influence of max_p on the packet loss rate is evident, since it is directly related to how harshly the RED scheme drops packets in order to discourage sources from sending too much data. The influence on stability is however indirect. Only varying w_q is not sufficient to avoid persistent oscillations of the instantaneous queue, whereas changing max_p helps in stabilizing the queue size.
2) w_q has a greater positive influence in terms of link utilization as compared to max_p. Moreover, the effect is prominent under conditions of heavy and moderate network traffic. The

drop in link utilization due to varying max_p can be countered by varying w_q in conjunction with max_p.

3) In mild scenarios, varying max_p alone can ensure a good performance in terms of loss percentage and link utilization.

V. CONCLUSIONS

In this paper, we have looked at the performance of RED scheme in relation to its operating parameters. We have identified the impact of varying max_p on link utilization and loss percentage. Previous papers on adaptive RED have focussed exclusively on tuning max_p, while ignoring the effects of w_q. We have studied the effect of varying w_q in conjunction with max_p and demonstrated how this can serve to counter the decrease in link utilization caused by varying max_p alone. We have suggested algorithms to adaptively vary w_q. We note that varying max_p has a greater influence on packet loss rate and queue length stability, while varying w_q controls the link utilization. Finally, it has been suggested that both max_p and w_q should be varied, in *harsh* and *moderate* congestion scenarios, to achieve an overall improvement in loss percentage and stability without adversely affecting the link utilization as compared to the default RED scheme.

In the end, we would like to reiterate the fact that RED is quite sensitive to its operating parameters. The issue of the performance of RED (adaptive or otherwise) in multiple congested networks is quite crucial and has not been investigated. We have explored active queue management strategies based on traffic prediction, and we feel that these algorithms are likely to fare better in multiple congested links and would be less sensitive to parameters. These issues are however beyond the scope of the current paper, and will be addressed elsewhere [11].

REFERENCES

[1] S. Floyd and V. Jacobson, *"Random Early Detection Gateways for Congestion Avoidance"*, ACM/IEEE Transactions on Networking, 1(4):397-413, August 1993.

[2] Wu-chang Feng, Dilip D. Kandlur, Debanjan Saha and Kang G. Shin, *"A Self-Configuring RED Gateway"*, Proceedings of IEEE INFOCOM, pp. 1320-1328, 1999.

[3] S. Keshav, *"An Engineering Approach to Computer Networking"*, Addison Wesley Longman Inc., 1999.

[4] Wu-chang Feng, Dilip D. Kandlur, Debanjan Saha and Kang G. Shin, *"Blue: A New Class of Active Queue Management Algorithms"*, University of Michigan Technical Report CSE-TR-387-99, April 1999.

[5] D. Lin and R. Morris, *"Dynamics of Random Early Detection"*, Proceedings of SIGCOMM 97, pp. 127-138, September 1997.

[6] Chris Hollot, Vishal Misra, Don Towsley and Wei-Bo Gong, *"A Control Theoretic Analysis of RED"*, Proceedings of INFOCOM 2001, 2001.

[7] Mikkel Christiansen, Kevin Jeffay, David Ott and F. Donelson Smith, *"Tuning RED for Web Traffic"*, Proceedings of ACM SIGCOMM 2000, Stockholm, Sweden, Aug.-Sep. 2000.

[8] Sally Floyd, Ramakrishna Gummadi, and Scott Shenker, *"Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management"*, `http://www.icir.org/floyd/papers/adaptiveRed.pdf`, August 2001.

[9] `http://www.aciri.org/floyd/REDparameters.txt`.

[10] `http://www.isi.edu/nsnam/ns/`

[11] Abhishek Jain, Rahul Verma and Abhay Karandikar, *"An Adaptive Prediction based Approach for Congestion Estimation in Active Queue Management (APACE)"*, `http://www.ee.iitb.ac.in/uma/~abhishek/research/apace.html`, June 2002.