Adaptive Prediction based Approach for Congestion Estimation (APACE) in Active Queue Management

Abhishek Jain, Abhay Karandikar and Rahul Verma

Information Networks Laboratory, Department of Electrical Engineering,
Indian Institute of Technology, Mumbai, India-400076
{*E-mail: karandi@ee.iitb.ac.in*}

Abstract

Active Queue Management (AQM) policies provide an early indication of incipient congestion to the sources. In this paper, we propose a new AQM policy called APACE. APACE stands for Adaptive Prediction based Approach for Congestion Estimation in Active Queue Management that predicts the instantaneous queue length at a future time instant using adaptive filtering techniques. We compare the performance of APACE with other existing AQM schemes in networks having both single and multiple bottleneck links. We show that APACE is able to control the oscillations in the instantaneous queue. We also demonstrate, through exhaustive simulations, that APACE performs well in terms of link utilization even in networks with multiple bottleneck links. Moreover, APACE is not very sensitive to parameter settings and adapts quickly to changes in traffic.

Key words: TCP, Congestion Control, Active Queue Management (AQM), Random Early Detection (RED).

1 Introduction

TCP (and its variants) remains the dominant end-to-end congestion control mechanism deployed in the Internet. The essence of this mechanism is that a TCP source adjusts its window size based on an implicit feedback about the congestion in the network. This implicit feedback is in the form of lack of receipt of acknowledgement from the receiver within a certain time out interval or the receipt of three duplicate acknowledgements. Either of these feedbacks is taken as an indication of packet loss. In a simple DropTail node, a packet gets dropped whenever the buffer is full. In networks with large Round Trip Times (RTT) and subsequently longer time out periods, the TCP congestion control has slower response

to a packet drop. Thus it is desirable that the sender be notified early so as to adjust its rate pre-emptively in order to avoid congestion in the bottleneck node. Active Queue Management (AQM) policies attempt to estimate the congestion at a node and signal the incipient congestion by dropping packet(s) before the buffer is full. A responsive congestion control strategy then reduces its transmission rate. This helps in avoiding further congestion and is expected to reduce the packet loss rate and keep the average queue size low. But if packets are dropped aggressively, then the capacity of the node may remain underutilized. An AQM policy thus has two components - one component estimates the congestion and another component takes the packet drop decision. The performance, therefore, depends upon how aggressive or conservative the estimate of congestion is and also on how aggressively the packets are dropped based on this estimate.

In this paper, we propose and analyze a new AQM strategy called APACE. The primary contribution of this paper is to propose an adaptive prediction based approach for active queue management that is simple to implement and not sensitive to parameter settings. We also attempt to give a general framework in terms of *operating points* for evaluating any AQM policy. We use this framework to compare APACE with other existing AQM schemes like RED [1], SRED [2], AVQ [3] and PAQM [4] in terms of Delay-Loss trade-off and Delay-Link utilization trade-off curves. These trade-off comparisons have been reported for the first time here in the literature. We have performed these experiments for networks having single as well as multiple bottleneck links. Our simulation results indicate that the APACE is able to effectively control the oscillations in the instantaneous queue and in that respect its performance is comparable to PAQM. Moreover, APACE scheme achieves higher link utilization and a lower packet loss for a given delay in networks with multiple bottleneck links than that of other AQM schemes including PAQM.

The rest of the paper is organized as follows. In Section 2, we discuss some of the related work in the area of active queue management. We then explain the concept of operating point in Section 3, the APACE algorithm in Section 4, discuss prediction accuracy in Section 5 and significance of various parameters in Section 6. We compare the performance of our scheme with RED, SRED, AVQ, and PAQM in Section 7 for single bottleneck link scenario and in Section 8 for a multiple bottleneck link scenario. We finally conclude the paper in Section 9.

2 Related Work

The RED scheme was initially described and analyzed in [1] with the main aim of providing "congestion avoidance" by dropping packets in anticipation of congestion. The performance of the RED algorithm depends significantly upon the setting of each of its parameters, i.e., w_q, max_p, min_{th} and max_{th}. Though significant progress has been made towards the understanding of tuning RED parameters, this problem has not yet been addressed satisfactorily. The difficulty in tuning the parameters of RED under different network conditions has limited its effectiveness. 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. Incorrectly tuned parameters may in fact cause RED to perform worse than that of Drop tail. In [5,6], the authors have questioned the benefits of RED by performing experiments on testbeds. They also argue that RED performs well only under single bottleneck gateways and heavy TCP traffic. These are also the cases for which most of the simulations have been performed and reported. The performance of RED and most other schemes under multiple bottleneck gateways has not yet been satisfactorily studied.

Various authors [7–9] have given guidelines and proposals for setting RED parameters or to adaptively vary them. In [10], Hollot et. al. have studied the problem of tuning RED parameters from a control theoretic stand point. The aim is to improve the throughput by controlling oscillations in the instantaneous queue. Feng et. al. [11] have proposed a mechanism for adaptively varying one of the RED parameters, max_p, with the aim of reducing the packet loss rates across congested links. Floyd et. al. in [12] discuss the modifications to the self-configuring RED algorithm [11] for tuning max_p adaptively. Their objective is to control the average queue length around a pre-decided target. The choice of the target queue size, left to the network operator, determines the trade-off between delay and link utilization. Controlling the average queue size, however, has a limited impact on regulating the packet loss rate. Balanced RED (BRED) [13] and Fair RED [14] aim to improve the fairness of RED by maintaining per-active-flow state information.

Adaptive Virtual Queue (AVQ) [3] decouples congestion measure from the performance measure. Stochastic Fair Blue (SFB) [15] attempts to enforce fairness among a large number of flows. It handles and rate limits non-responsive flows effectively using an extremely small amount of state information. CHOKe (CHOose and Keep for responsive flows and CHOose and Kill for unresponsive flows) [16] also aims to ensure fairness to each of the flows that share the outgoing link.

Predictive AQM (PAQM) [4] tries to exploit traffic predictability in the calculation of the packet dropping probability. The authors have shown that the correlation structure present in long-range dependent traffic can be used to accurately predict the future traffic. PAQM enables the link capacity to be fully utilized without incurring excessive packet loss by stabilizing the instantaneous queue to a desired level. This scheme is an attractive AQM policy but is very compute intensive. We propose an AQM scheme which is comparable to PAQM in terms of queue stability but has other attractive features like higher link utilization with low packet loss and delay under multiple bottleneck links.

3 Operating Point

An AQM policy can be used to control the performance metrics such as link utilization, average queuing delay and packet loss rate. An AQM policy should give the freedom to specify the required performance metrics and should be able to meet the requirements to its best. In other words, one should be able to specify the *operating point* that one wants to achieve given a particular network scenario. The operating point, for instance can be

specified in terms of link utilization, average queuing delay and packet loss rate that one wants to achieve. We denote such an operating point by (t^*, d^*, l^*), where t^* is the target link utilization, d^* is the target average queuing delay and l^* denotes the target packet loss rate. In order to keep our representation simple and be able to visualize the graphs in two dimension, in this paper we define operating points as (t^*, d^*) and $(l^*, d^*,)$. It is possible that a desired operating point might not be achievable. In that case, the aim of the AQM policy should be to approach this operating point as closely as possible.

In case of a Drop Tail gateway, there is only one operating point for a given network scenario since the packets are dropped only when the buffer is full. In RED and APACE, the parameters can be suitably adjusted resulting in greater choices of operating points. Setting the appropriate parameters is difficult in case of RED, but as will be illustrated later, in APACE we need to vary only one parameter to achieve a given operating point.

4 Proposed Scheme

In APACE, we estimate the congestion by predicting the instantaneous queue length at a future time instant. This estimate is based on the queue lengths at the previous packet arrivals. The decision to drop any packet is based on the predicted value of the instantaneous queue length rather than the average queue length, as in the case of RED. As will be shown later, this makes the scheme more responsive especially in scenarios with changing network conditions. We now explain the APACE scheme in detail.

4.1 Predicting the Instantaneous Queue

We predict the instantaneous queue length using the Normalized Least Mean Square (NLMS) algorithm [17]. Our simulation results show that the NLMS predictor can be used to obtain a good estimate of the instantaneous queue length under a large set of network scenarios (different kinds of sources, topologies etc.). Moreover, the algorithm requires only a few iterations to converge and adapts well under changing network scenarios.

The instantaneous queue length prediction is made at every packet arrival. Let M denote the order of the NLMS filter used for prediction, $q(n)$ the instantaneous queue length at the nth packet arrival and $\bar{q}(n)$ a Mx1 vector of the instantaneous queue lengths of the past M packet arrivals. The instantaneous queue length after N_0 packet arrivals from the n^{th} packet arrival instant is predicted based on $\bar{q}(n)$. We call N_0 the prediction parameter. We denote the predicted queue length by $\hat{q}(n + N_0)$. $\hat{q}(n + N_0)$ is calculated as follows:

$$\hat{q}(n + N_0) = \bar{w}_q^T(n) * \bar{q}(n) \tag{1}$$

In the above equation, $\bar{w}_q(n)$ denotes a Mx1 weight vector. These weights are updated

dynamically based on the error between the predicted and the actual queue length. The error, $e(n)$ in the prediction is computed as

$$e(n+N_0) = q(n+N_0) - \bar{w}_q^T(n) * \bar{q}(n) \tag{2}$$

The queue weights are updated using the following equation:

$$\bar{w}_q(n+1) = \bar{w}_q(n) + \mu(n) * \bar{q}(n) * e(n) \tag{3}$$

In the NLMS algorithm, $\mu(n)$ is calculated using the following equation:

$$\mu(n) = \frac{\mu_0}{1 + \bar{q}^T(n)\bar{q}(n)} \tag{4}$$

The queue weights are initially set to fixed values and later updated using the above equations. Typically, the weights are initially set to 0. In our simulations, μ_0 has been set to 0.01. A small value of μ_0 implies guaranteed convergence of the NLMS algorithm though at a slower rate. A large value of μ_0 increases the rate of convergence but may cause the algorithm to diverge.

4.2 Taking a Packet Drop Decision

As stated earlier, the decision to drop the incoming packet is based on the predicted value of the instantaneous queue length. The incoming packet is dropped with a probability p that is calculated based on $\hat{q}(n+N_0)$. The algorithm for dropping the incoming packet(s) is illustrated in Figure 1. Let B denote the maximum buffer size. If $\hat{q}(n+N_0) < \alpha * B$, no packet is dropped (where α is a positive constant less than 1). If $\hat{q}(n+N_0) > B$, every incoming packet is dropped. If $\alpha * B \leq \hat{q}(n+N_0) \leq B$, the incoming packet is dropped with a probability p, which is a function of the predicted queue size. For the purpose of simulations, we vary the probability p linearly from 0 at αB to max_p at B. The motivation behind linearly increasing the packet dropping probability is to make the scheme more aggressive as the predicted queue length increases. The packet dropping probability, p can thus be expressed as:

$$p \leftarrow \frac{max_p(\hat{q}(n+N_0) - \alpha * B)}{(1-\alpha) * B} \tag{5}$$

On every packet arrival

- Predict instantaneous queue size (after N_0 packet arrivals):
  $\hat{q}(n+N_0) \leftarrow \bar{w}_q^T(n) * \bar{q}(n)$

- Calculate packet dropping probability p:

  · If $\alpha * B \leq \hat{q}(n+N_0) \leq B$

  $p \leftarrow \frac{max_p(\hat{q}(n+N_0)-\alpha*B)}{(1-\alpha)*B}$

  · else if $\hat{q}(n+N_0) \geq B$
  $p = 1$

  · else if $\hat{q}(n+N_0) \leq \alpha * B$
  $p = 0$

- Update queue weights using the NLMS algorithm

Fig. 1. The APACE algorithm

5 Prediction Accuracy

The first issue that needs to be addressed is whether the NLMS algorithm can predict the instantaneous queue length accurately. We test the performance of NLMS algorithm under different network loads. The simulations have been performed using the network simulator, **ns v2.1b8a** [18]. The network topology shown in Figure 2 has been used for simulations.

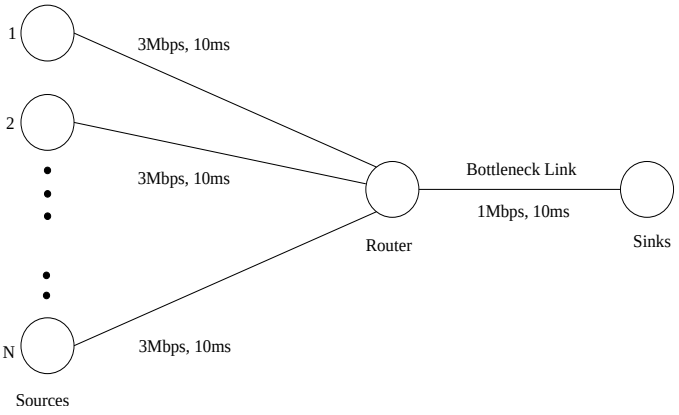

Fig. 2. Network Topology

Figure 3 illustrates the mean square error in the actual and the predicted value of the instantaneous queue size (learning curve). The plot has been obtained for 25 TCP sources. The packet loss rate is relatively high $\approx (5-10)\%$. The mean square error has been averaged

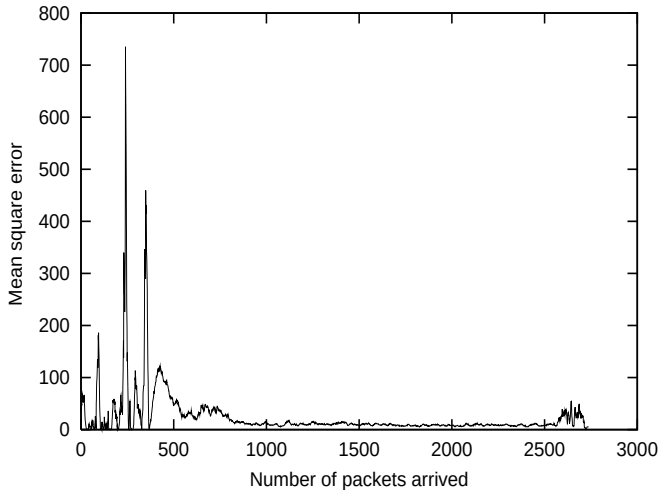

Fig. 3. Learning curve

over 100 sample paths. As can be seen from the figure, the NLMS algorithm converges quickly and is able to predict the instantaneous queue to a reasonable accuracy (residual error [1] ≈ 8 square packets). We get similar learning curves even for fluctuating network loads (described in Section 7), though the prediction in case of sustained heavy traffic is better as compared to other scenarios with lower or fluctuating network loads. This is because under conditions of heavy congestion, the aggregate incoming traffic characteristics is more predictable. Moreover, even under fluctuating network loads, the algorithm is able to adapt to the network conditions in a few iterations only. The error in the queue prediction for the same is shown in Figure 4.

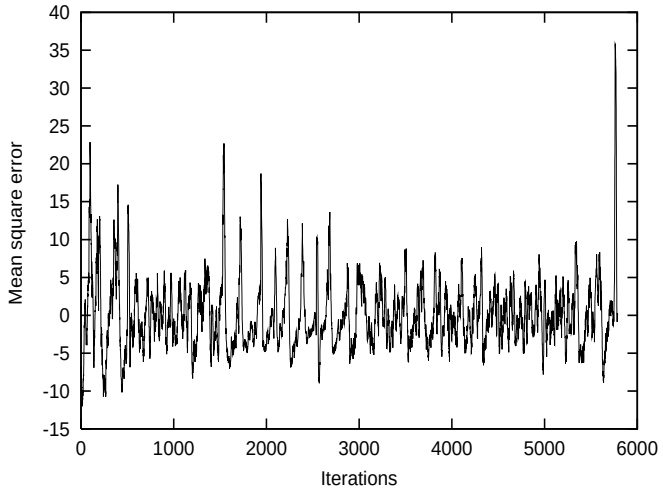

Fig. 4. Error for fluctuating traffic scenario

Based on the above discussion and simulation results, we can conclude that the NLMS algorithm does a good prediction of the instantaneous queue for various traffic scenarios.

[1] Residual error refers to the steady state mean square error in predicting the instantaneous queue.

6 Significance of Various Parameters

The various parameters of the scheme are listed in Table 1. Our simulation results indicate

Table 1
APACE parameters

Parameter	Function
M	Order of the filter
N_0	Prediction parameter for the filter
max_p	Maximum dropping probability
α	Decides the lower threshold below which no incoming packet(s) is dropped

that the order of the filter M does not have a significant impact on the performance of the scheme. However, for large values of $M(\geq 50)$, the NLMS algorithm does not converge. This is because the NLMS algorithm typically takes $20M$ iterations to converge and during this period, the traffic arriving at the queue might vary substantially. Moreover, the computational complexity of the scheme increases with increasing M. Therefore, it is advisable to keep a low value for the order of the filter. We suggest that $M \approx (5-20)$ is a good value for the order of the filter and the performance of the scheme is not sensitive to M in this range. For the purposes of simulations, we have chosen $M = 10$.

The prediction parameter N_0 decides the trade-off between the accuracy of the prediction and how early the prediction is made, i.e., a lower value of N_0 means that the prediction is made over a relatively shorter time scale. In such a scenario, the prediction error is typically low. On the other hand, by taking a large value of N_0, the prediction is made over a longer time scale but the prediction error might be large. The value of N_0 should ideally depend on the round trip time (RTT) as well. N_0 should be large enough so that the effect of dropping a packet results in an early congestion notification to the source. This implies that for large RTTs, one should keep a large value of N_0, though this might result in a greater residual error in prediction. Keeping a small value of N_0 in scenarios with large RTTs, might result in a delay in the congestion estimation (since the prediction instant is not far enough in future) as well as a delay in the congestion notification (owing to a large RTT) to the source.

Our exhaustive simulation results also illustrate that N_0 does not have a strong bearing on the performance (in terms of packet loss rate, average queuing delay and link utilization) of the scheme. We have reported simulations results for $M = 10$ and $N_0 = 15$. An extensive simulations study to determine the effects of these parameters can be found in [19].

The parameter max_p governs how aggressively the packets are being dropped, based on the predicted value of the instantaneous queue length and α determines the buffer occupancy at which we should start dropping packets. Figure 5 illustrates the variation of the average queuing delay as a function of max_p in a harsh scenario (25 TCP sources and $5-10\%$ packet

loss rate). The different plots, as indicated, correspond to different values of α. The plots show that the average queuing delay is a non-increasing function of max_p. Moreover, for relatively high values of α (≥ 0.5), the delay stabilizes above a certain value of max_p (≈ 0.2). For low values of α, the delay keeps decreasing even for high values of max_p. In a mild scenario (5 TCP sources and $1-2\%$ packet loss rate) (see Figure 6), even for low values of α, the average queuing delay stabilizes beyond $max_p = 0.1$.

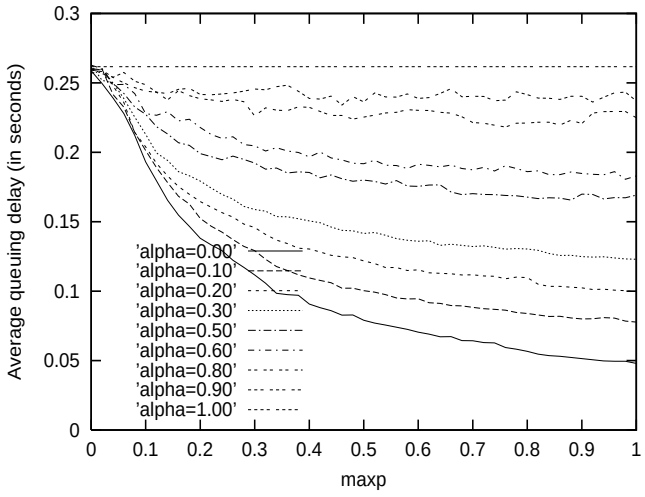

Fig. 5. Average queuing delay vs. max_p (different α), harsh scenario

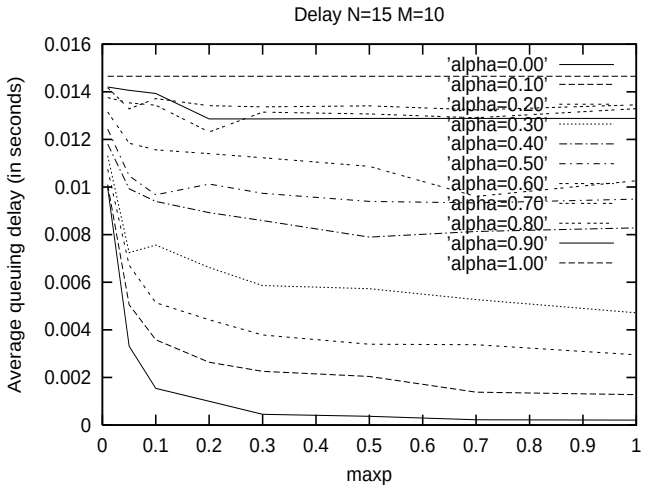

Fig. 6. Average queuing delay vs. max_p (different α), mild scenario

Figure 7 illustrates the variation of the packet loss rate against max_p in a harsh scenario. The plots show that the packet loss rate increases with increasing max_p. This is intuitive because a higher value of max_p results in a more aggressive dropping of packets leading to a greater packet loss rate. Moreover, the increase is more prominent for low values of α. For large values of α (> 0.5), the packet loss rate stabilizes beyond $max_p \approx 0.2$. However, in a mild scenario (Figure 8) the packet loss rate is independent of max_p for a large range of α ($\alpha > 0.2$).

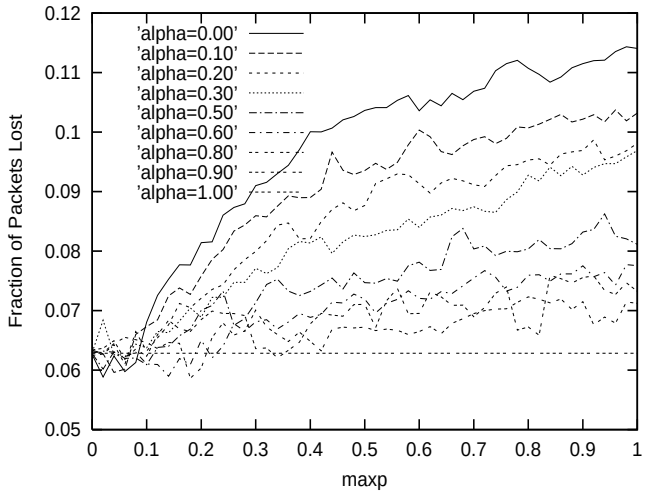

Fig. 7. Packet loss rate vs. max_p (different α), harsh scenario

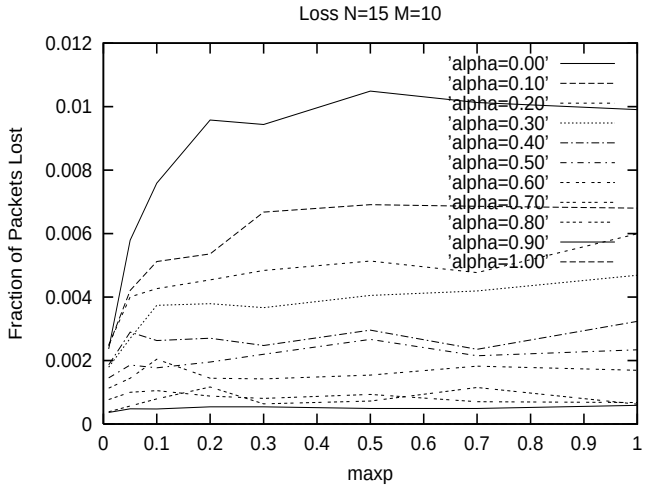

Fig. 8. Packet loss rate vs. max_p (different α), mild scenario

The effects of other parameters on the performance have been explained in detail in [19]. Extensive simulations by varying all the parameters indicate that the scheme gives similar performance under a wide range of parameter settings. The scheme adapts itself to the network conditions well, thus reducing the importance of initial parameter settings. We suggest $max_p = 0.2$, $\alpha = 0.3$, $N_0 = 15$, $M = 10$ as the default values.

7 Comparison with other Queuing Strategies

In this section, we compare the performance of APACE with other queuing schemes like RED, SRED, PAQM and AVQ. The network topology with single bottleneck link shown in Figure 2 is used for performing the simulations. The results for multiple bottleneck scenario are explained in the next section. The number of TCP sources, N, is varied to achieve

different incoming traffic loads. Packet size has been fixed at 500 bytes. The buffer size at the router is of 20 packets for SRED and 50 for all others. The reason for choosing 20 is the fact that SRED always tries to keep the buffer close to full. The results (except for the instantaneous queue) have been averaged over 20 sample paths.

7.1 Instantaneous Queue Length Stability

We first address the issue of instantaneous queue stability. In our simulation, 40 TCP sources are switched on randomly in the first two seconds and the simulation is performed for 40 seconds. Figure 9 illustrates the instantaneous queue size at the gateway for various queuing strategies. Instantaneous queue is measured in terms of number of packets in the queue throughout the paper. We have used $w_q = 0.002$, $min_{th} = 5$, $max_{th} = 15$, $max_p = 0.1$ for RED and the default parameters for APACE, i.e., $max_p = 0.2$, $\alpha = 0.3$, $N_0 = 15$, $M = 10$. The parameters for SRED are $M = 1000$, $\alpha = 1/M$ and $p_{max} = 0.15$ and for AVQ, $\gamma = 0.98$, $\alpha = 0.10$. We have chosen $Q_{opt} = 20$ for PAQM as the queue in APACE is stable at 20. The meaning of these parameters have been explained in respective papers [2–4]. Note that the meanings of M and α in APACE are different from those in SRED and AVQ.

As is evident from the plots, RED is unable to control the oscillations in the instantaneous queue, while APACE and PAQM provide reasonable stability to the instantaneous queue. Moreover, even under steady heavy traffic, the instantaneous queue in RED becomes empty frequently (as shown in Figure 9) leading to severe underutilization of the link. In addition, the average queue size in RED deviates significantly from the instantaneous queue. This motivates us to look for better indicators of congestion than the average queue length. The decision to drop a packet based on the average queue size also introduces delay in the estimation of congestion as the learning is slower. As a result, RED might take a wrong decision regarding a packet drop. SRED has problems of global synchronization similar to that of DropTail since it always keeps its buffer close to full.

We next perform the simulation under fluctuating network loads. We switch on 40 sources within a small interval of time. After about 10 seconds of the simulation, 36 of these sources are switched off resulting in a drastic decrease in the incoming traffic. At about 20 seconds from the start of the simulation, 20 new TCP sources are switched on resulting in a sudden increase in the incoming traffic and 10 seconds later additional 16 TCP sources are switched on. These 40 TCP sources run till 40 seconds from the start where the simulation is terminated. Even under such fluctuating network loads, the NLMS algorithm adapts very well and is able to predict the instantaneous queue accurately. The error in prediction of the instantaneous queue is plotted in Figure 4. We note that at points where there is a sudden change in the incoming traffic, the prediction error increases. However, the NLMS algorithm is able to converge and predict the instantaneous queue accurately and quickly.

The instantaneous queues are shown in Figure 10. The instantaneous queues are kept again around 20 for the sake of comparison. Conclusions similar to above can be drawn from it.

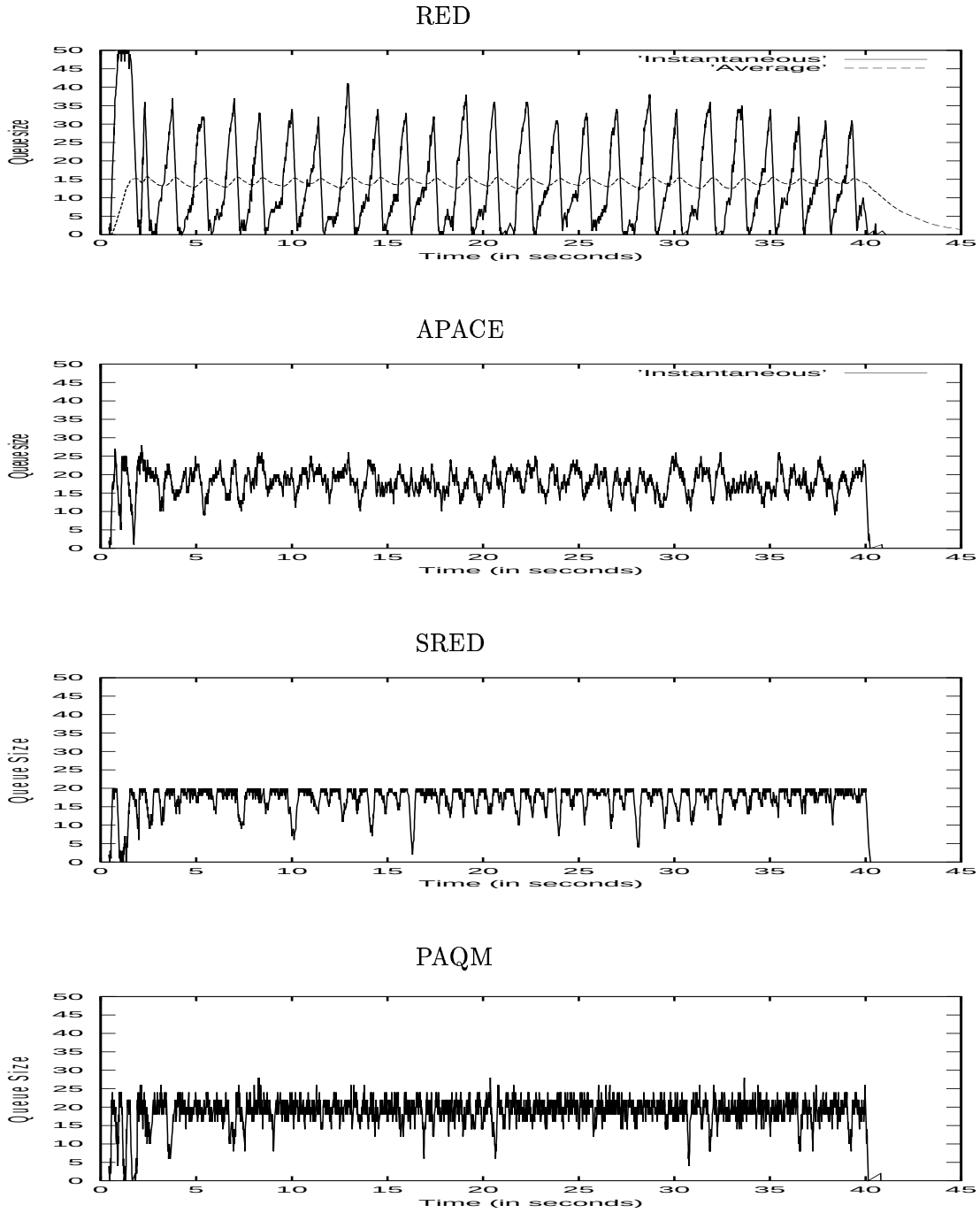

Fig. 9. Instantaneous queue at RED, APACE, SRED and PAQM routers under heavy traffic conditions

APACE is indeed able to adapt well to the changes in network conditions and in maintaining the stability of the instantaneous queue.

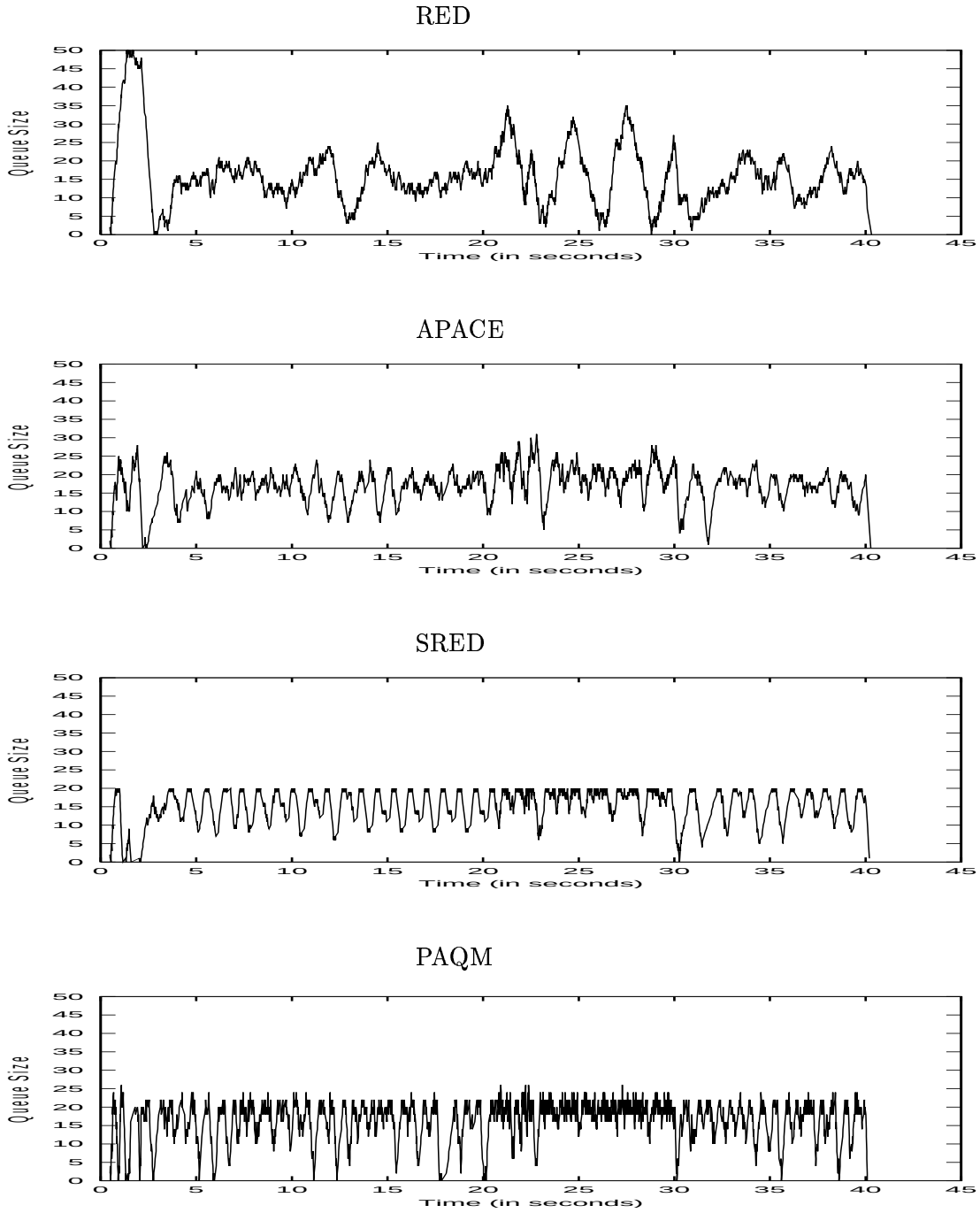

Fig. 10. Instantaneous queue at RED, APACE, SRED and PAQM routers under fluctuating traffic conditions

7.2 Link utilization

Figure 11 shows the number of packets transmitted successfully at the bottleneck node as the number of TCP connections is increased. Link utilization is measured as the total

number of packets transmitted successfully by the router. We observe that APACE is able to successfully transmit more packets than any other scheme that we have simulated.

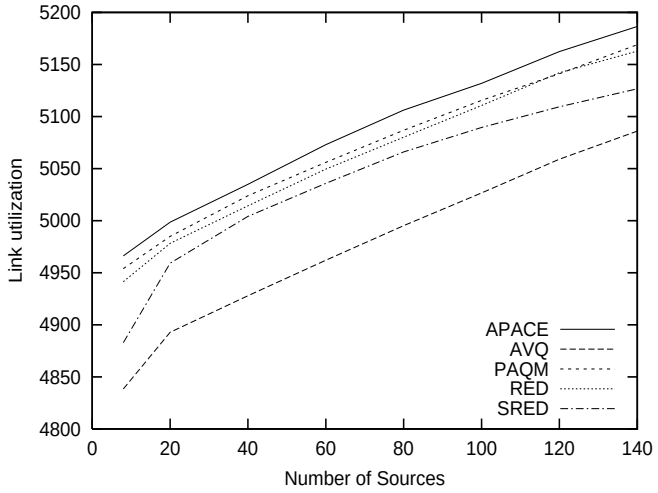

Fig. 11. Link utilization vs. number of sources in a single bottleneck scenario

7.3 Packet loss rate

Figure 12 illustrates the fraction of packets dropped at the bottleneck node as the number of TCP connections is increased. As can be seen from the figure, APACE shows a consistent improvement in the packet loss rate. This improvement is more prominent under heavy network loads (larger number of TCP sources) because under mild congestion scenarios, the packet loss, as such, is quite low.

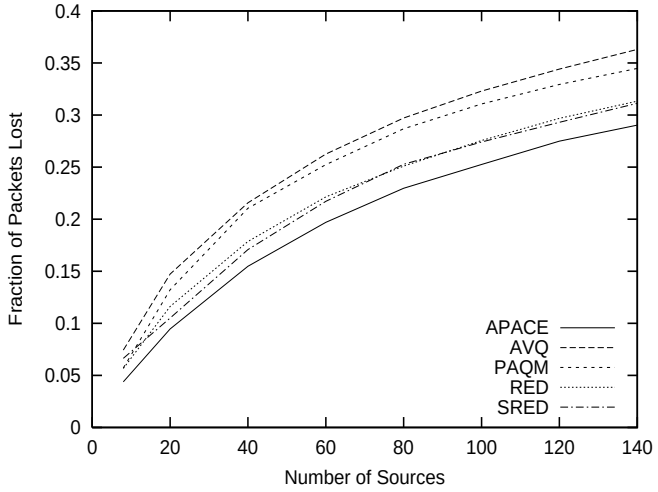

Fig. 12. Fraction of packets lost vs. number of sources in a single bottleneck scenario

It should be noted that it is unfair to compare APACE and RED by fixing any one set of parameters. In fact, the above statement holds true for any scheme whose performance needs to be compared with RED. One might achieve an entirely different performance for some other setting of RED parameters. Moreover, using only one performance metric to compare any scheme with RED is also not entirely correct because finally there is a trade-off between various performance metrics such as delay-link utilization or delay-loss. Also we need to take into consideration the effects of various parameters on the performance metrics.

Hence, we now compare the performance of APACE scheme with RED in terms of the delay-link utilization trade-off curves (refer Figures 13, 14) and delay-loss trade-off curves (refer Figures 15, 16). The buffer size is 250 packets. The simulations have been performed with 80 TCP sources. We have measured delay in seconds throughout.

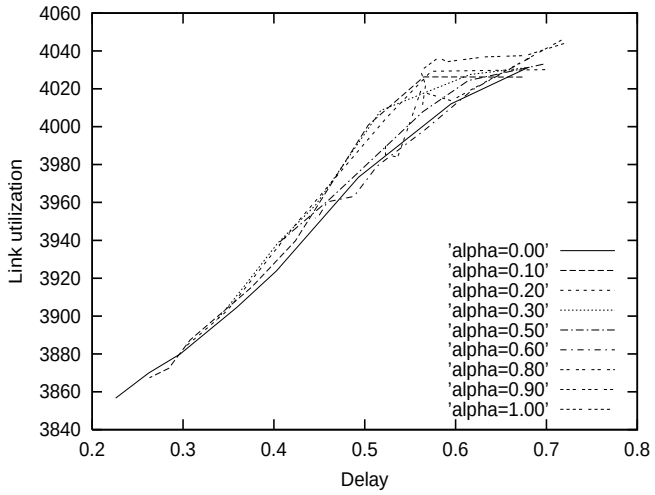

Fig. 13. Delay-link utilization trade-off curves for single bottleneck scenario APACE

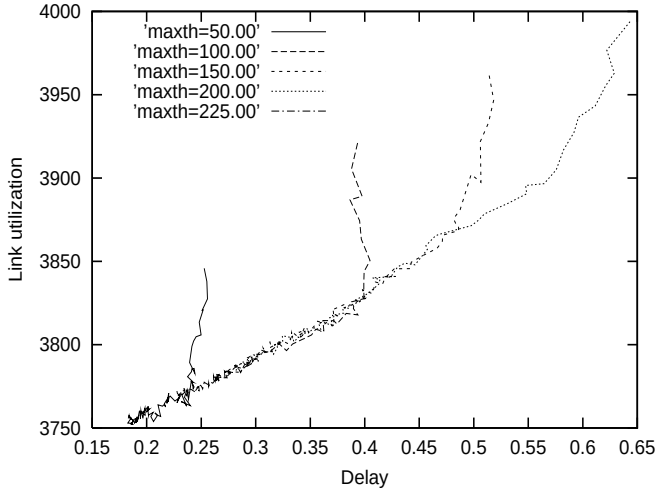

Fig. 14. Delay-link utilization trade-off curves for single bottleneck scenario with RED

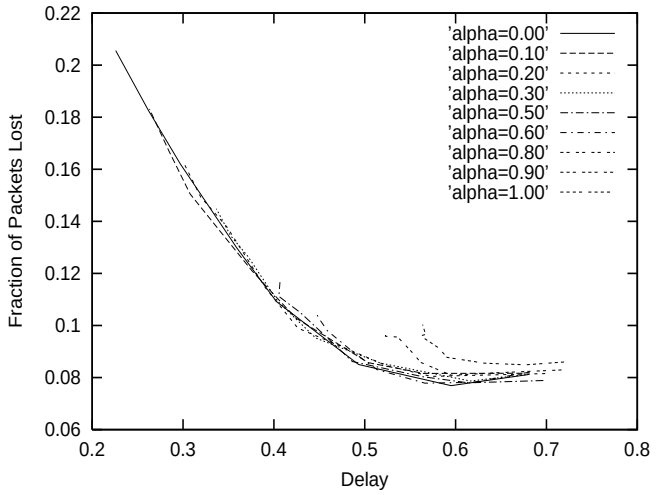

Fig. 15. Delay-loss for trade-off curves for single bottleneck scenario with APACE

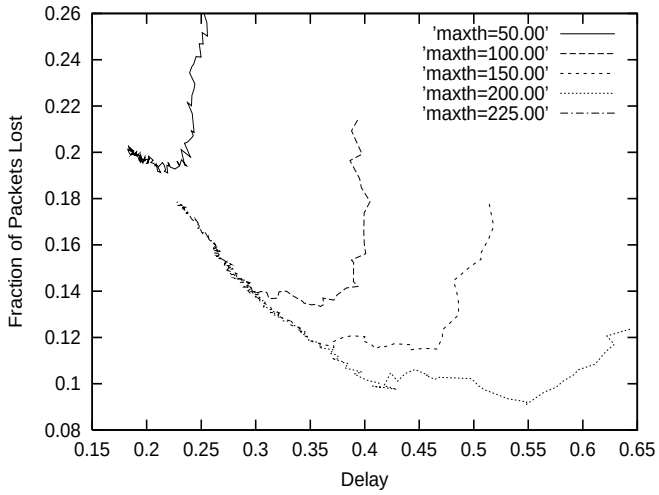

Fig. 16. Delay-loss trade-off curves for single bottleneck scenario with RED

Each curve corresponds to a different value of max_{th} (α) for RED (APACE). The extreme right point on each curve corresponds to $max_p = 0$ and as we move along the curve, we get points corresponding to higher values of max_p. It is worth noting that the plots for different values of α are close to each other and in particular, the plot with $\alpha = 0$, encompasses the complete range of the delay-link utilization trade-off plane covered by the other plots (for different values of α). Therefore, by setting $\alpha = 0$ and varying only max_p, one can span all the achievable delay-link utilization trade-off operating points. An operating point below the delay-link utilization trade-off curve is an *achievable* operating point, while the one above it is *unachievable*. By an achievable operating point, we mean that given a particular delay-link-utilization pair that one wants to achieve, an operating point with a higher link-utilization and lower delay can be achieved.

As can be seen from Figures 13 and 14, APACE gives a better link utilization and also lower delay than that of RED. Also any operating point that is achievable by RED can be achieved

by APACE as well by a suitable setting of parameters. Moreover, for the APACE scheme (refer Figures 13, 15) one can cover the entire range of operating points by keeping $\alpha = 0$ and varying max_p only. The above observations are also true when the buffer size is 50 packets. This, however, can not be said for RED and both the parameters, max_p and max_{th} need to be varied (Figure 14) in order to achieve a certain operating point. This makes APACE scheme easy to adapt to network conditions. By varying only max_p *adaptively* based on the network traffic, we can achieve a better AQM strategy than the existing ones.

We have compared the performance with only RED here, mainly to illustrate the significance of the concept of trade-off curves. Trade-off comparisons with other schemes are reported in the next section for multiple bottleneck link scenarios. From the above we can conclude that the APACE gives a better link utilization with lower delay and only one parameter max_p needs to be varied to achieve any given feasible operating point.

8 Performance under Multiple Bottleneck Links

We now compare the performance of the various AQM schemes in networks having multiple congested bottleneck links. The network topology is shown in Figure 17. There are "N" TCP sources connected to router 1 and a cross traffic of 25 TCP sources each flows from router 2 to router 3 and from router 4 to router 5 respectively. Packet size has been fixed at 500 bytes and buffer size at 50 packets. The results (except for instantaneous queue) have been averaged over 20 sample paths.

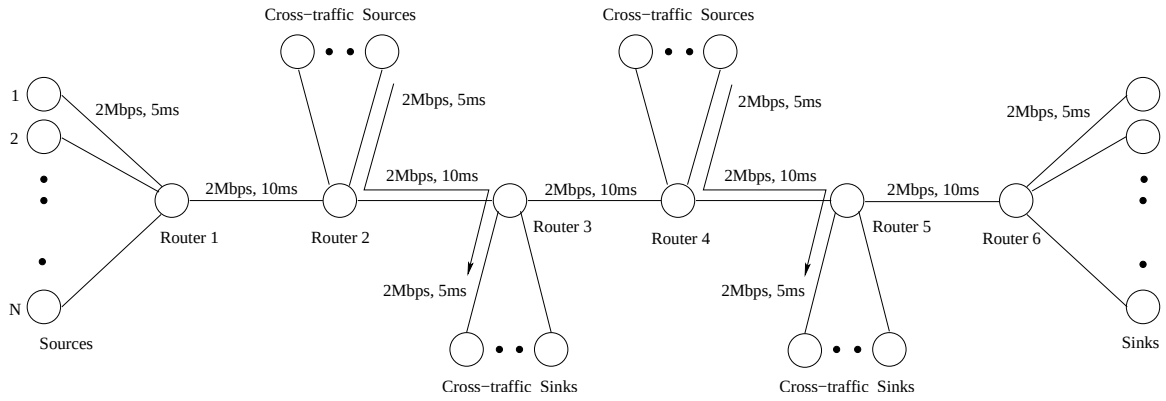

Fig. 17. Network topology for multiple bottleneck links

Routers 2 and 4 are the most congested nodes and hence show similar behavior. We have chosen router 2 for detailed study. We verify our earlier observation by extensive simulations that values of N_0 and M do not affect the performance significantly. The results are in fact similar to those for the single bottleneck case in terms of dependence on N_0 and M. We choose the same parameter settings as before, except that $Q_{opt} = 40$ for PAQM and a buffer size of 40 packets for SRED at all the five nodes.

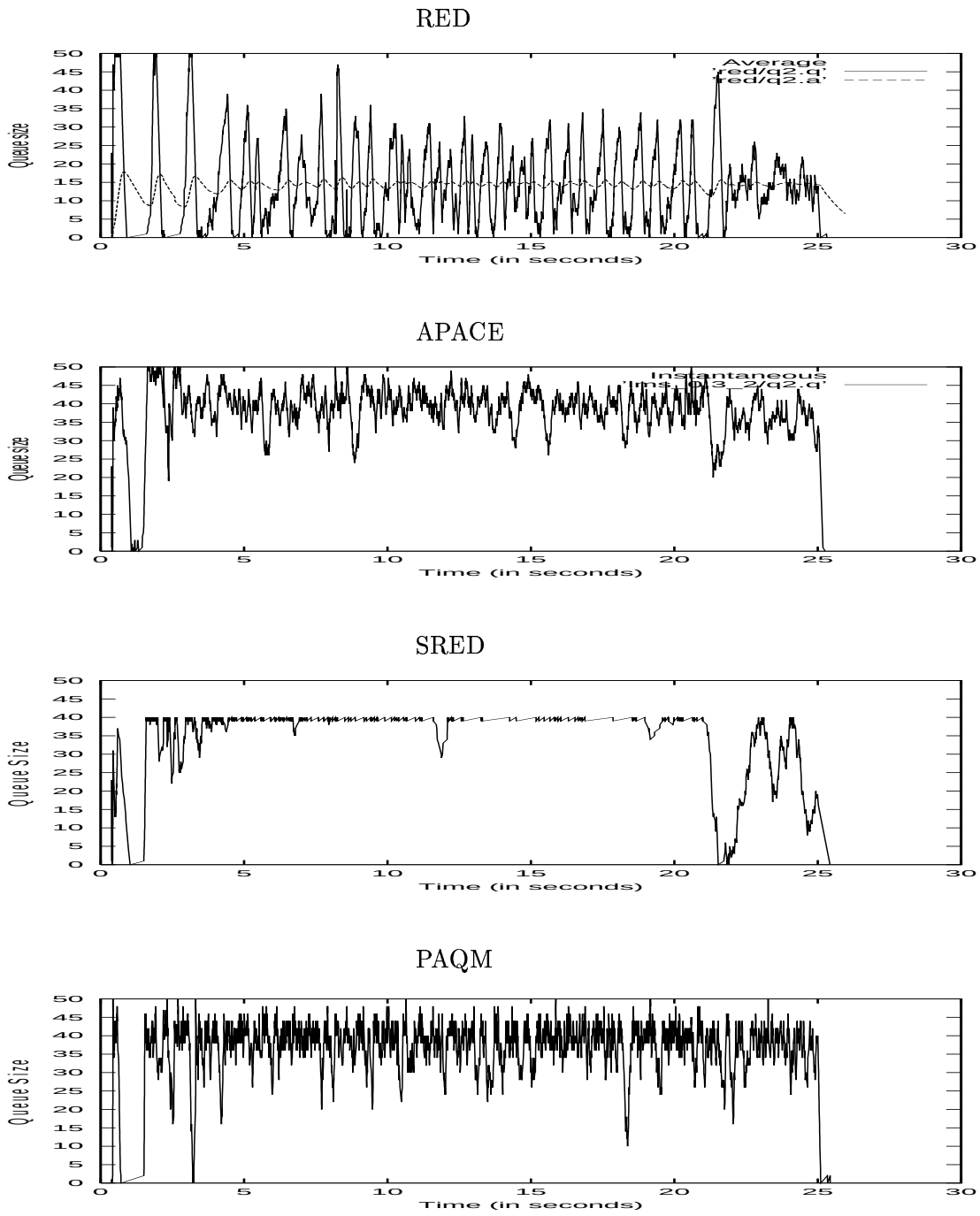

Fig. 18. Instantaneous queue at RED, APACE, SRED and PAQM routers in a multiple bottleneck scenario

The instantaneous queue length results are similar to those obtained for single bottleneck link. The queue occupancy is around 40 packets instead of 20 previously. We observe that APACE is able to keep the instantaneous queue stable in agreement to our observations for

the single bottleneck case. This implies that the packet delay remains almost constant under the APACE scheme.

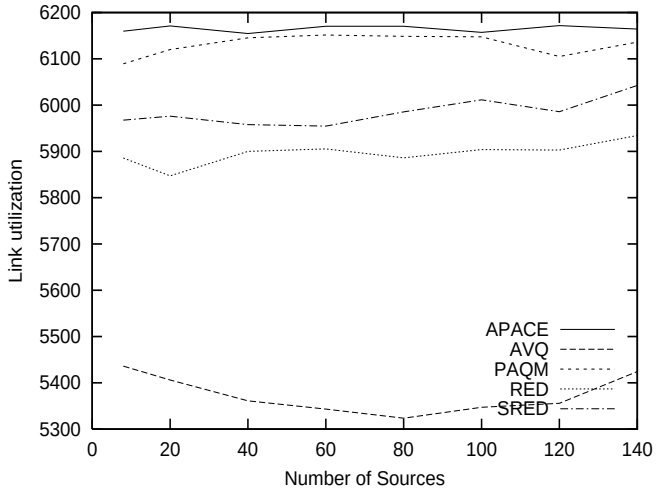

Fig. 19. Link utilization vs. number of sources in a multiple bottleneck scenario

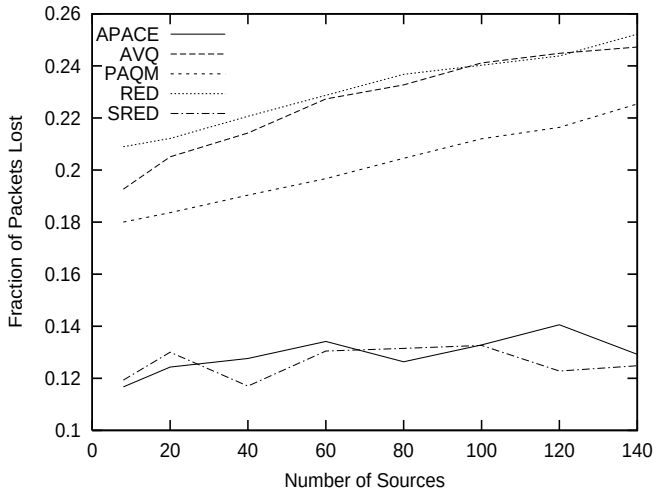

Fig. 20. Fraction of packets lost vs. number of sources in a multiple bottleneck scenario

In addition APACE also gives better link utilization and lower packet loss as shown in Figures 19 and 20. Link utilization and fraction of packets lost are illustrated in Figures 19 and 20 respectively, as the number of TCP connections is increased. We observe that in networks with multiple bottleneck links, APACE is able to achieve high link utilization with low packet loss rate and a stable queue that keeps the delay bounded.

8.2 Trade-off Curves

The delay-link utilization and delay-loss trade-off curves for the various schemes are shown in Figures 21 and 22 respectively. The curves for APACE correspond to $\alpha = 0$ and varying max_p. Though APACE has better curves for other values of α, we plot the curves for $\alpha = 0$

while comparing it with others to illustrate the fact that we can indeed fix α at 0 or some other small value and vary only max_p and still get performance better than that of other schemes. For RED, the curves correspond to $max_{th} = 15$ and varying max_p. Q_{opt} in PAQM and buffer size in SRED is varying from 3 to 50 as we move from left to right and α is varying from 0.05 to 0.99 in AVQ.

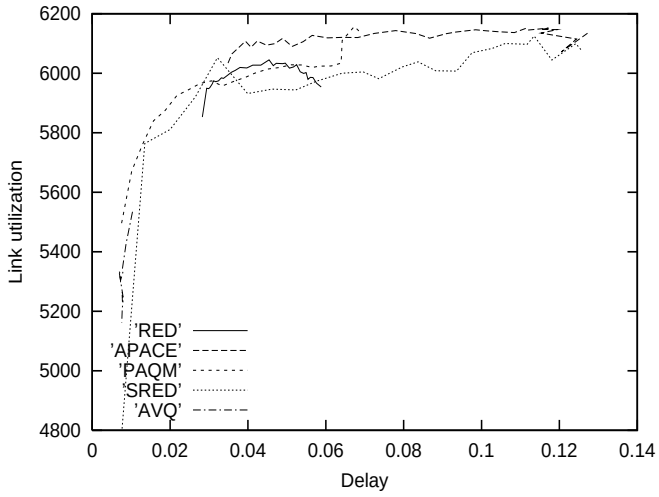

Fig. 21. Delay-link utilization trade-off curves for multiple bottleneck scenario

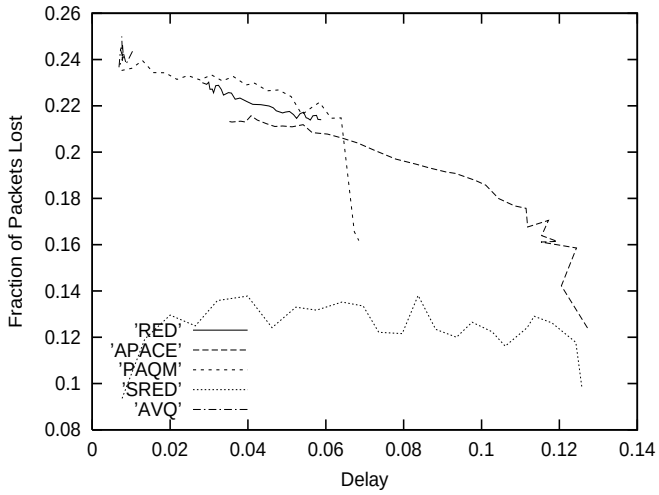

Fig. 22. Delay-loss trade-off curves for multiple bottleneck scenario

We observe that for a given delay, APACE achieves the highest link utilization at a much lower packet loss rate. Though the trade-off curves for SRED look better, it is associated with other problems like global synchronization owing to its frequent overflow of buffer. In addition link utilization in APACE remains almost constant at a high value indicating that it is quite independent of its parameter settings and hence the network operator can focus more on the delay and packet loss rate without worrying much about link utilization.

Note that the drop in the link utilization-delay trade-off curve in Figure 21 for $max_{th} = 15$

which is the default value for RED indicates that we would achieve lower link utilization and higher packet delays at least for the traffic scenario considered here (which is common) for higher values of max_p. This suggests that varying max_p adaptively may sometimes worsen the performance of RED under multiple bottleneck links.

8.3 Stability of Link Utilization

Figure 23 shows the link utilization of RED at router 2 as its parameters are varied. For RED we have taken 4 values for max_{th}, i.e., 15, 30, 45, 50. For every value of max_{th}, we have varied max_p from 0.00 to 1.00. For these different values, we have plotted the link utilization on the Y-axis. Similarly for APACE, we have varied max_p from 0.00 to 1.00 and α from 0.0 to 1.0. The link utilization is plotted on the Y-axis of Figure 24. We observe that the link utilization remains fairly constant for the different values of its parameters as compared to RED. In fact, the initial part of Figure 23 corresponds to $max_{th} = 15$, which is also its default value. The average packet delay and packet drop rate vary in both the schemes. Given this flexibility in parameter adjustment for link utilization in addition to the fact that the instantaneous queue remains fairly constant and its level of occupancy can be controlled by varying only max_p may be useful to give delay guarantees and achieve better link utilization.

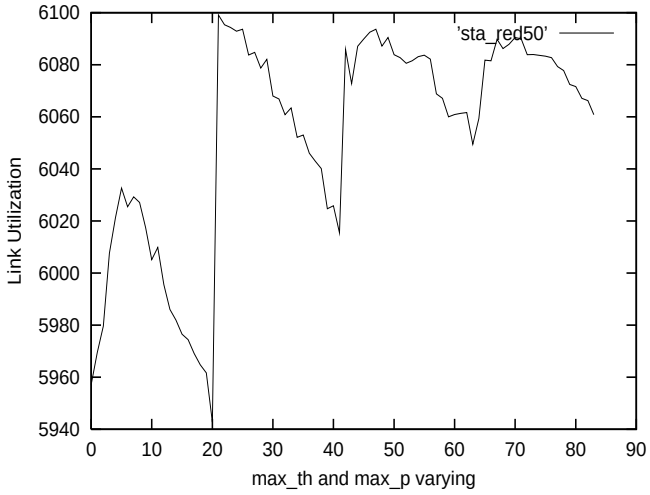

Fig. 23. Link utilization of RED with various parameter settings

9 Conclusions

In this paper, we have presented a new AQM scheme based on (predicted) instantaneous queue length. The main contributions of the paper can be summarized as follows-

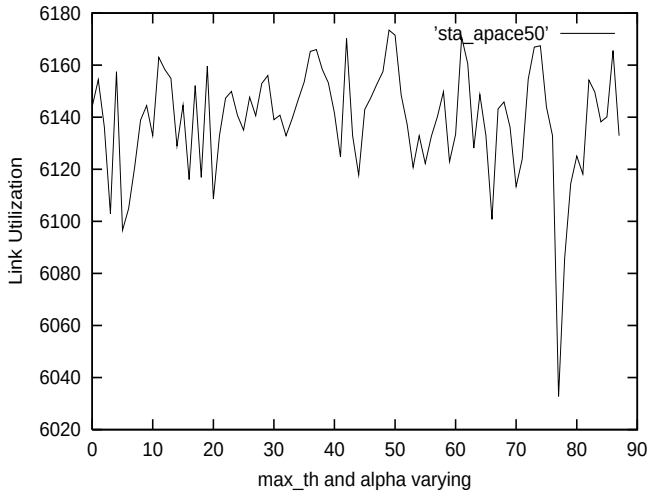

Fig. 24. Link utilization of APACE with various parameter settings

- We have shown that NLMS adaptive algorithm can indeed be used to predict the instantaneous queue length.
- An AQM scheme based on this predicted queue length is able to achieve the stability of the queue. The scheme is also simple to implement.
- Since there is a complex inter-play between various performance metrics of an AQM scheme, we have argued that these AQM schemes must be compared in terms of Delay-Loss and Delay-Link utilization trade-off curves. We have performed exhaustive simulation experiments and compared AQM schemes in terms of these trade-off. It has been argued [6] that schemes like RED perform well in simulation environments of single bottleneck but do not perform well in multiple bottleneck scenario. We have, therefore, performed our experiemnts for both these scenarios. The results of these trade-off comparisons for schemes like RED, SRED, AVQ and PAQM were also not available earlier.

Our results demonstrate that APACE is able to achieve the instantaneous queue stability comparable to PAQM. Moreover, APACE gives higher link utilization and a much lower packet loss rate as compared to schemes like PAQM. In this paper, we have not addressed the issue of bandwidth sharing between adaptive flows like TCP and non-adaptive flows like UDP. Typically, per-flow state information like in [14] is required to achieve fariness. Such schemes are therefore not scalable. In [20], the authors have argued that AQM scheme when used in combination with fair scheduling algorithm can provide better isolation and fairness. The issue of APACE in conjunction with fair scheduling needs to be investigated further. One limitation of APACE is its overhead in terms of computation of the predicted value of the instantaneous queue at every packet arrival. A feasible solution is to predict the queue at periodic intervals. However, there will be a trade-off between the prediction interval and the accuracy of the prediction. This issue requires more investigation.

References

[1] Sally Floyd and Van Jacobson. Random Early Detection Gateways for Congestion Avoidance. *ACM/IEEE Transactions on NetworkingACM/IEEE Transactions on Networking*, 1(4):397–413, August 1993.

[2] Teunis J. Ott and T. V. Lakshman and Larry H. Wong. SRED: Stabilized RED. In *Proceedings of IEEE INFOCOM*, pages 1346–1355, 1999.

[3] Srisankar Kunniyur and R. Srikant. Analysis and Design of an Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management. In *Proceedings of ACM SIGCOMM*, pages 123–134, August 2001.

[4] Yuan Gao, Guanghui He, and Jennifer C. Hou. On Exploiting Traffic Predictability in Active Queue Management. In *Proceedings of IEEE INFOCOM*, pages 1415–1424, 2000.

[5] T. Bonald , M. May and J. Bolot. Analytic Evaluation of RED Performance. In *Proceedings of IEEE INFOCOM*, 2000.

[6] Martin May, Jean Bolot, Christophe Diot and Bryan Lyles. Reasons not to deploy RED. In *Seventh International Workshop on Quality of Service*, June 1999.

[7] Sally Floyd. RED: Discussions of Setting Parameters. http://www.aciri.org/floyd/REDparameters.txt.

[8] Van Jacobson, K. Nichols, and K. Poduri. RED in a Different Light. http://www.cnaf.infn.it/~ferrari/papers/ispn/red_light_9_30.pdf, September 1999.

[9] H. Ohsaki and M. Murata. Steady state Analysis of the RED Gateway: Stability, Transient Behaviour, and Parameter Setting. *IEICE Transcations on Communications*, E85-B(1), January 2002.

[10] C. V. Hollot, Vishal Misra, Don Towsley, and Wei-Bo Gong. A Control Theoretic Analysis of RED. In *Proceedings of IEEE INFOCOM*, pages 1510–1519, 2001.

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

[12] 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.

[13] F. Anjum, and L. Tassiulas. Fair Bandwidth Sharing among Adaptive and Non-Adaptive Flows in the Internet . In *Proceedings of IEEE INFOCOM*, pages 1412–1420, 1999.

[14] Don Ling, Robert Morris. Dynamics of Random Early Detection . In *Proceedings of ACM SIGCOMM*, 1997.

[15] Wu-chang Feng, Dilip D. Kandlur, Debanjan Saha and Kang G. Shin. Stochastic Fair Blue: A Queue Management Algorithm for Enforcing Fairness. In *Proceedings of IEEE INFOCOM*, pages 1520–1529, 2001.

[16] Rong Pan, Balaji Prabhakar and Konstantios Psounis. CHOKE, A Stateless Active Queue Management Scheme for Approximating Fair Bandwidth Allocation . In *Proceedings of IEEE INFOCOM*, pages 942–951, 2000.

[17] Simon Haykin. *Adaptive Filter Theory*. Third Edition, Prentice-Hall, 1996.

[18] Network SImulator. http://www.isi.edu/nsnam/ns/.

[19] Abhishek Jain. Congestion Management and Bandwidth Allocation for Best Effort Traffic in Packet Switched Networks. Master's thesis, Department of Electrical Engineering, Indian Institute of Technology, Bombay, Mumbai, India, June 2003.

[20] B. Suter, T. V. Lakshman, D. Stiliadis and A. K. Choudhary. Buffer Management Schemes for supporting TCP in Giagbit Routers with Per-flow Queueing. *IEEE Journal on Selected Areas in Communications*, 17(6), June 1999.