Telecommun Syst (2017) 65:281–295
DOI 10.1007/s11235-016-0229-4

Adaptive mean queue size and its rate of change: queue management with random dropping

Karmeshu[1] · **Sanjeev Patel**[1] · **Shalabh Bhatnagar**[2]

Published online: 14 September 2016

Abstract The random early detection active queue management (AQM) scheme uses the average queue size to calculate the dropping probability in terms of minimum and maximum thresholds. The effect of heavy load enhances the frequency of crossing the maximum threshold value resulting in frequent dropping of the packets. An adaptive queue management with random dropping algorithm is proposed which incorporates information not just about the average queue size but also the rate of change of the same. Introducing an adaptively changing threshold level that falls in between lower and upper thresholds, our algorithm demonstrates that these additional features significantly improve the system performance in terms of throughput, average queue size, utilization and queuing delay in relation to the existing AQM algorithms.

Keywords Active queue management (AQM) · Rate of change of average queue size · Throughput · Queuing delay · AQMRD · Traffic control

1 Introduction

Active queue management is the most effective network assisted algorithm to control the congestion at the routers. There are several AQM algorithms proposed in the literature viz. drop-tail, random early detection (RED), random early marking (REM) etc. RED has been able to enhance the throughput by using dropping function in terms of the average queue size [1]. A desirable feature of the algorithm is to reduce the loss rate in the presence of random traffic characteristics. Another algorithm REM introduces rate mismatch as well as queue mismatch to yield high throughput or achieve low loss-rate [2]. Queue mismatch is the difference between a target queue length and current queue length. The rate mismatch is the difference between the rate at which a sender sends the data and the rate at which a receiver receives the data and thus it is directly associated with data rate. Accordingly, REM differs from RED as it uses a different measure for congestion. This congestion measure or price is updated based on both mismatches observed for queue and rate. Similarly, REM computes the prices for each link and then calculates their sum to determine the end-to-end marking probability. The latter probability increases with higher congestion measures or link prices [2]. A new algorithm BLUE improves the performance by reducing the loss rate of the packets by modifying the dropping function based on loss events [3]. A separate first-come-first-served (FCFS) queue is required in fairness-queuing (FQ) scheme for each conversation. The queues are serviced in a round-robin fashion so as to allocate equal bandwidth to each queue. Stochastic fairness queuing (SFQ) is proposed to avoid infeasible computational requirements for high-speed networks [4]. In the case of large number of queues as compared to the number of conversations, a high probability is provided for each conversation being assigned to its own queue. The flow or pair of source and destination receives less than the allocated shared bandwidth when two conversations collide. SFQ provides a mechanism so that collided conversations for one slot are very unlikely to collide during the next [4].

✉ Shalabh Bhatnagar
shalabh@csa.iisc.ernet.in

[1] School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi 110067, India

[2] Department of Computer Science and Automation, Indian Institute of Science, Bangalore 560012, India

A variant of RED has been proposed in [5] to resolve the limitations observed with traditional RED [6]. Refining RED, Wang et al. [7] have enhanced the throughput by keeping the average queue size below the threshold value to avoid dropping of the packets. This has been realized by first adapting the queue weight parameter and thereafter the values of the maximum probability (max_p) parameter are chosen to stabilize the queue size. A limitation with RED is that packets are discarded even when the queue size is lower than the threshold value. Feng et. al. [8] have addressed the issue of unnecessary dropping of packets by requiring additional information on instantaneous queue size. The important aspects for the stabilization of queue in the RED gateway are discussed in [9–11]. A robust optimization technique for RED is analyzed in [12]. This technique is independent of technology, model and protocols used. A detailed comparative study of AQM algorithms can be found in [13]. In RED, the queue size varies according to congestion level which leads to unpredictable queuing delay. A major issue with traditional RED has been the setting of the parameters and their tuning in order to achieve good performance [6]. Adaptive RED achieves significant control resulting in improved throughput using dynamic adaptation of the max_p parameter. Feng et al. [14] have proposed a three-section RED (TRED) to overcome this issue particularly for heavy loads by dividing the queue size interval ($max_{th} - min_{th}$) into three equal sections where max_{th} and min_{th} are the maximum and minimum thresholds respectively. In TRED, dropping probability is calculated according to the dropping function used in the three different sections. Nonlinear dropping functions are used in lower and upper sections of the queue size interval. In addition, the middle interval uses a linear dropping function. TRED is able to improve the throughput at low loads and maintains low delays at high loads [14].

The problem of parameter setting in TRED is not addressed as in RED. Further, Adaptive RED also does not show good performance under high load conditions. In the case of proportional controller, it is found to suffer from a limitation under certain situations in which it is infeasible to implement. The instability is found in proportional controller if an operating point of p lies between max_p and 1 for such network conditions which result in oscillations of queue length. This leads to increase in the queuing delay and this issue arises due to the coupling between the average queue size and the dropping probability. The Proportional Integrator (PI) controller has been proposed to decouple both the average queue size and the dropping probability [15,16]. The PI controller is implemented with the nonlinear TCP dynamic by introducing the role of the queue's operating point. Wu et al. [17] have proposed an interactive mobile streaming scheme which has the capability to maintain a certain level of service quality in the presence of dynamic network environments. Most of the AQM schemes have been studied for wired networks. Adaptive Optimized Proportional Controller (AOPC) is another novel AQM scheme based on controlling the queue parameter [18]. The dropping probability is updated for a very small time interval upon each packet arrival in AOPC. AOPC measures packet loss ratio for larger intervals by introducing load estimator for the network. TCP load is estimated with the help of packet loss ratio and then TCP/AOPC feedback is optimized according to the second-order system model [18]. It makes AOPC stabilize the queue length very close to the target queue length using an optimized second-order system model. Chavan et al. [19] have, however, discussed the robust AQM for wireless networks where a challenging problem results from fading and the resulting change in bandwidth allocation [20]. Chavan et al. [19] have designed a better queue controller by tuning the operating points offline. An analytical model for bursty and correlated traffic to compute performance measures such as end-to-end delay, loss probability, and throughput is also proposed in [21] in the presence of hybrid wireless networks. This analytical model is also applicable for bursty and correlated traffic and has been tested on OMNeT++ simulator.

One of the serious limitations of the aformentioned algorithms is that they are based on information regarding the mean queue size alone. However, in reality the rate of change of queue size on account of non-stationary heavy traffic may provide much deeper insights into the growth dynamics of queue build up. Our work aims to incorporate information both about the average queue size at any time and its rate of change. It is useful to point out that the proposed algorithm based on AQMRD is not related to REM. In the context of REM, the quantity of interest is the difference between arrival rate of packets and the link capacity, whereas in AQMRD, it is the rate of change of the average queue size in relation to the mid-threshold. In fact, prior work completely ignores this second-order dynamics that our work aims to capture. Our work makes our system akin to a dynamical system where not just the velocity but also information on acceleration is captured through the system measurements and this new information is also used to control the system dynamics. We observe that this (latter) information indeed helps obtain better control over the system.

This paper comprises of five sections. Section 2 discusses a queue rate based model which is introduced to overcome the problems observed in the existing algorithms. The simulations and discussions are presented in Sect. 3. Section 4 evaluates the performance of our proposed scheme with existing schemes using ns-2 simulator. In the last section, we provide the concluding remarks.

2 System model

2.1 Queue-rate based model

A new approach which incorporates both queue size and its rate of change would better characterize the evolutionary dynamics of queue size build up. The dropping function is a key variable which affects the throughput or the loss-rate in the AQM algorithm. The dropping function in RED is based on the assumption that the input traffic characteristics do not change much with time. In contrast to RED, we have introduced a new parameter $davg$, corresponding to the rate of change of the average queue size in addition to the average queue size (avg) itself. The proposed AQMRD approach calculates the quantities avg and $davg$ according to

$$avg(t+1) = (1-w_q)\, avg(t) + w_q\, q(t+1), \tag{1}$$

$$davg(t+1) = (1-w_q)\, davg(t) + w_q(q(t+1) - q(t)), \tag{2}$$

where w_q is a weight parameter, $avg(t)$ and $davg(t)$ are the average queue size and the rate of change of the average queue size respectively at time t. Here $q(t)$ denotes the instantaneous queue length at time t. We increment time by one unit after the arrival of every packet. Accordingly, the information about rate of change of the average queue size would reflect the traffic characteristics to achieve better congestion control at the routers. A positive value of the rate of change of avg indicates that the queue size attains the threshold rapidly whereas a negative rate of change indicates a slow-down in the process of reaching the threshold.

In order to prevent packet drop at the gateways, our aim is to modify the dropping function so that unused buffer space remains available to accommodate the new arriving packets. This is achieved by noting that a positive value of $davg$ results in a reduction in the number of received packets. RED has been found to perform well in the case of moderate traffic. One of the deficiencies of RED is that it is hard to configure and is sensitive to traffic load. Our proposed algorithm is able to capture the changing nature of non-stationary traffic. In the RED gateway, the average queue size is compared to two thresholds, maximum threshold and minimum threshold. However, the effect of non-stationary heavy traffic results in the increase in frequency of crossing or reaching the max_{th} and thus resulting in more packets being dropped. In order to address this limitation, we have modified the RED algorithm by introducing a new variable, viz., mid-threshold (mid_{th}) lying between min_{th} and max_{th}, which adapts to rapidly changing temporal variation of avg. The proposed extended framework results in qualitatively much improved values of the performance metrics. To avoid reduction in the number of received packets, we have applied a more aggressive dropping function by varying mid_{th} towards min_{th}.

This allows significant reduction in the number of discarded packets and is achieved when a gateway marks the packet more aggressively before avg reaches the max_{th}. As a negative rate of change will slow down the process of reaching max_{th}, it will not adversely affect the dropping of the packets. However, a positive rate of change is likely to force the system to enter into an undesirable state resulting in dropping of several packets. The introduction of a mid_{th} takes care of such an adverse situation. The proposed queue-rate based model allows more unused space to be left in the buffer, thus reducing the number of times avg crosses the maximum threshold. Consequently, it results in an increase in the number of received packets by dynamically updating mid_{th} depending on the value of $davg$.

2.2 Proposed algorithm: AQMRD

In the light of the significant role played by $davg$, we have proposed a new queue-rate based algorithm for the gateways. The AQMRD gateway performs different decisions for mid_{th} according to (3). Here, mid_{th} is updated as follows:

$$mid_{th} = \begin{cases} mid_{th} + 1 & \text{if } davg < 0, \\ mid_{th} - 1 & \text{if } davg > 0, \\ mid_{th} & \text{if } davg = 0. \end{cases} \tag{3}$$

The dropping probability function is calculated as follows:

$$p_b = \begin{cases} p_1 & \text{if } davg > 0, \\ p_2 & \text{if } davg \le 0, \end{cases} \tag{4a}$$

where p_1 and p_2 are given by

$$p_1 = \frac{avg - min_{th}}{mid_{th} - min_{th}} max_p, \tag{4b}$$

$$p_2 = \frac{avg - min_{th}}{max_{th} - min_{th}} max_p. \tag{4c}$$

For a positive value of $davg$, the AQMRD gateway starts dropping packets when avg crosses the threshold mid_{th} rather than max_{th}. The AQMRD gateway for positive $davg$ calculates the dropping probability according to

$$p_b = \begin{cases} 0 & \text{if } min_{th} > avg, \\ \frac{avg - min_{th}}{mid_{th} - min_{th}} max_p & \text{if } min_{th} \le avg < mid_{th}, \\ 1 & \text{if } mid_{th} \le avg, \end{cases} \tag{5}$$

where mid_{th} is a variable which varies between min_{th} and max_{th} according to (3). In the case of a non-positive value of $davg$, the dropping probability is computed as

$$p_b = \begin{cases} 0 & \text{if } min_{th} > avg, \\ \frac{avg - min_{th}}{max_{th} - min_{th}} max_p & \text{if } min_{th} \leq avg < max_{th}, \\ 1 & \text{if } max_{th} \leq avg. \end{cases} \quad (6)$$

The Proposed AQMRD Algorithm:

synchronization. AQMRD's dropping function is shown in Fig. 1. The advantage of the proposed approach is that it does not suffer from the phenomenon of global synchronization due to the dynamic nature of mid_{th}. The advantage of AQMRD is that it achieves a better throughput for heavy traffic with the help of prior information regarding the change in

```
Initialization:
        count ← −1
for each packet arrival
   calculate the average queue size avg and rate of change of average queue size davg
       if min_th ⩽ avg < max_th
           increment count
      if davg > 0
          decrement the mid_th
          if avg < mid_th
            calculate probability p_1
          else
             the probability p_1 set to 1
         calculate probability p_b using p_1
         update the dropping probability
                                   p_a ← p_b / (1 − count.p_b)
         mark the arriving packet with probability p_a
      else
          increment the mid_th
         calculate probability p_2
          calculate probability p_b using p_2
          update the dropping probability
                                   p_a ← p_b / (1 − count.p_b)
         mark the arriving packet with probability p_a
        if gateway mark the arriving packet
          count ← 0
else if max_th ⩽ avg
   mark the arriving packets with probability one
   count ← −1
```

The min_{th} is not kept too low in AQMRD gateway to avoid underutilization of the bandwidth, particularly in view of fluctuations arising on account of heavy and non-stationary traffic. Depending upon the permissible delay, the threshold max_{th} is set. If the difference between max_{th} and min_{th} is small then average queue size is likely to reach the maximum queue size frequently as seen in the drop-tail scheme. Accordingly, to circumvent the problem, we choose mid_{th} so that it varies according to

$$mid_{th} = x.min_{th} \text{ where } x \in [1, 3]. \quad (7)$$

In case of a positive value of $davg$, the difference between mid_{th} and min_{th} should be sufficient enough to avoid global

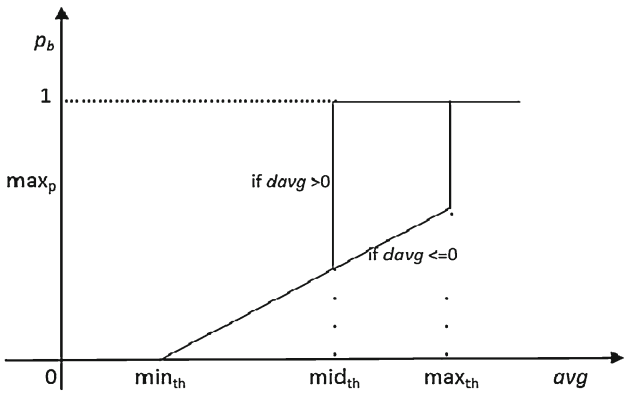

Fig. 1 AQMRD's packet dropping function

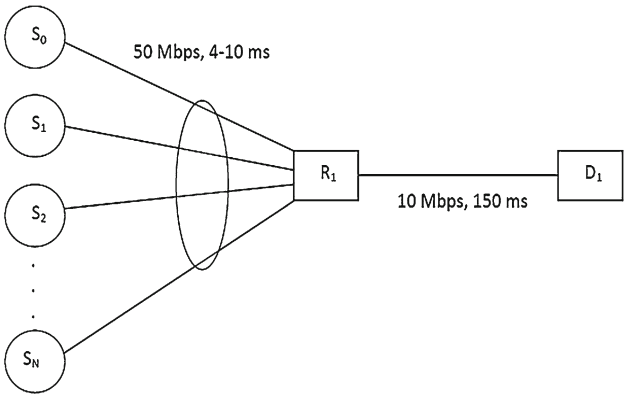

Fig. 2 Network topology

average queue size and the rate of change of the same. This dynamic adaptation of mid_{th} results in an adaptive nature of queue management and which in turn results in improvement of the performance irrespective of the parameter setting. The next section presents a comparative study of our scheme with existing AQM schemes using the ns-2 network simulator.

3 Simulations and discussions

3.1 The simulation setup

We consider the network topology as shown in Fig. 2. The gateway/router parameters for our scheme used in the experiments are set as $w_q = 0.002$, $max_p = 0.1$, and $max_{th} = 48$ packets. The queue parameter min_{th} is set at one-third of max_{th} in each simulation to achieve good performance [8]. A gateway has a buffer size of 64 packets and each FTP source sends a packet with a maximum packet size of 1000-bytes until the congestion control window allows sending of the packets. The choice of parameter values in AQMRD is dictated by the fact that the values should be similar to the ones commonly used in REM, PI, SFQ and Adaptive-RED. This will aid in fair comparison of the results of AQMRD with the other existing AQM schemes. We now provide some values of the parameters used in REM, SFQ, and PI. The parameter settings for REM are $\phi = 1.001$, $\alpha = 0.1$, $\gamma = 0.001$ and $b^* = 20$ packets. These values are taken from Athuraliya et al. [2]. The parameters ϕ, α, and γ are constant parameters and b^* is the target queue length. The constant ϕ is always taken to be greater than one. Noting from Athuraliya et al. [2], the constant α provides a trade off between utilization and queuing delay during the transient phase, while γ controls the responsiveness of REM to changes in network conditions. For the above set of values, it is found that REM's performance is good. Mckenney [4] sets the default values in SFQ to $maxqueue = 40$ and $buckets = 16$. In the case of PI, Hollot et al [15] have suggested the values of the PI coefficients and sampling frequency as $a = 0.00001822$, $b = 0.00001816$, $w = 170$ respectively, and the target queue as $q_{ref} = 50$. Some of the AQM schemes are sensitive towards their parameter settings. For each scheme there are some parameters which are responsible for controlling the congestion. The purpose of maintaining the parameters to their prescribed values in the literature (cf. [2,4,5,8,15]) is to provide fair comparisons of the algorithms in the literature while keeping the parameter settings as prescribed therein, with our AQMRD algorithm. The parameters for AQMRD have been set to similar values as the other algorithms in the literature in order to have a fair comparison with the other algorithms as well as to get an idea of the efficacy of our scheme.

For performing simulation, we assume random propagation delay varying from 4 to 10 ms between the FTP sources and router. Each simulation is run for a duration of 100 s. We choose maximum probability $max_p = 0.1$ to check the efficacy of the proposed scheme with respect to the parameter setting. Due to the adaptive nature of our proposed scheme as discussed later, AQMRD achieves good performance even for a high maximum probability as $max_p = 0.1$, indicating that our scheme has low sensitivity towards the max_p parameter because at a low value of max_p, our scheme achieves good performance. We set the delay-bandwidth product at around 200 packets which equals the congestion window size [8].

3.2 AQMRD's parameter settings

In our experiments, we have simulated settings with 25, 50, 75 and 100 FTP sources, respectively. These correspond to low, moderate, high and very high traffic load conditions. For our first set of experiments, we have simulated 25 FTP sources for the network topology shown in Fig. 2. We show in Fig. 3 the behavior of both average and instantaneous

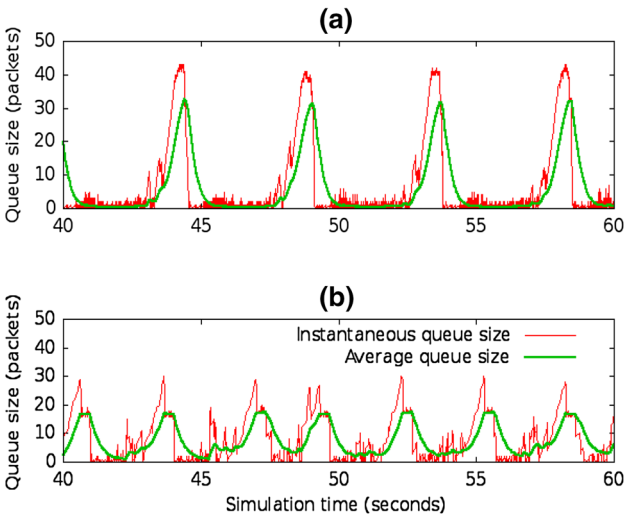

Fig. 3 AQMRD versus RED: comparative changes in q and avg for $N = 25$. **a** RED, **b** AQMRD (Color figure online)

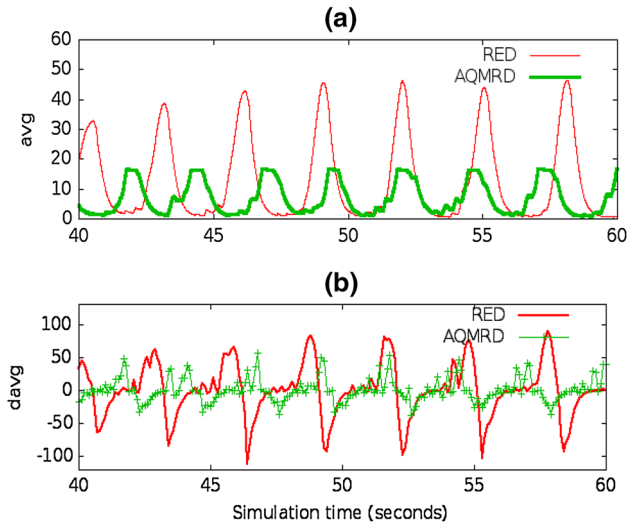

Fig. 4 AQMRD versus RED: comparative changes in avg and $davg$ for $N = 50$. **a** Average queue size, **b** rate of change of avg (Color figure online)

queue sizes in RED and AQMRD gateways for these 25 FTP sources. This shows that the AQMRD gateway suppresses the fluctuations in the average queue size and instantaneous queue size by more than what is observed in the RED gateway. This lower value of average queue size reduces the queuing delay in our scheme.

For moderate traffic load of $N = 50$ FTP sources, changes in both average queue size and rate of change of the average queue size come into picture. Fig. 4 shows a comparison between RED and AQMRD for avg and $davg$. In contrast to RED, it indicates that the avg and $davg$ both are stabilized sufficiently by AQMRD. The mid_{th} is adapted according to traffic load because it depends on the value of $davg$ that differs for different traffic loads. The parameter mid_{th} is increased by one for negative value of $davg$ and decreased by one for positive value of $davg$. There is no change in mid_{th} if $davg$ equals zero. The reason behind changing the mid_{th} is to adjust the dropping probability subject to enhancing the performance parameters for our scheme. A negative value of $davg$ indicates that the average queue size decreases with the rate of $davg$ and a positive value of $davg$ indicates that the average queue size increases with the rate of $davg$. If the average queue size increases then it indicates that more packets are coming into the queue. To avoid larger estimates of dropped packets, AQMRD's aim is to increase the dropping probability by decreasing the mid_{th} value. Similarly, for negative value of $davg$ the parameter mid_{th} increases just to reduce the dropping probability.

It is important to highlight the mechanism for setting the parameters for simulation of AQMRD algorithm. There are two new parameters viz. mid_{th} and $davg$ and remaining parameters are the same as in RED or other variants of RED such as MRED, Adaptive RED, TRED etc. The mid_{th} is a variable which varies from min_{th} to max_{th} depending on the value of $davg$. Feng et al [8] have suggested that min_{th} should be kept to a level of 1/3 of max_{th} to get good performance. The principle behind setting of the parameters w_q, min_{th}, max_{th} and max_p is given for instance in Floyd et al. [1] and Feng et al. [8]. The setting of w_q is crucial as it reflects how fast the gateway detects congestion. A low value of w_q does not detect initial congestion. The maximum value of w_q as given in Floyd et al. [1] is 0.0042. Accordingly we set $w_q = 0.002$ in our simulation. It may be noted that both max_{th} and min_{th} depend on the average queue size. The utilization declines as min_{th} is lowered and thus min_{th} cannot be made very small. However, on the other hand, max_{th} is always less than the buffer size. We keep $(max_{th} - min_{th})$ large for achieving reduced number of drops. If avg tends to max_{th}, the dropping probability tends to max_p. A higher value of $(max_{th} - min_{th})$ indicates a smaller slope for the dropping function resulting in lower number of packet drops. For high congestion, max_p is set to 0.1 because in a congested link, a higher dropping probability helps in avoiding the overflow. For the AQMRD gateway, we have shown in Fig. 5 the variation of mid_{th} with respect to the simulation time for $N = 25$ and $N = 50$. Depending on the value of $davg$, mid_{th} stabilizes between min_{th} and max_{th}. Simulation results show that mid_{th} oscillates between $min_{th} = 16$ and $max_{th} = 48$, and stabilization varies from min_{th} to max_{th}.

4 Performance evaluation

4.1 Simulations under different scenarios

We note in our setting that input traffic increases when the number of FTP sources reaches around 25 for a chosen

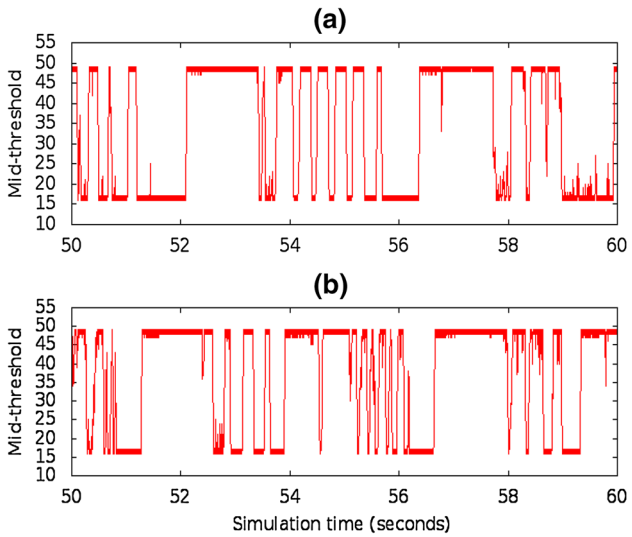

Fig. 5 Comparison of mid_{th} between $N = 25$ versus $N = 50$. **a** $N = 25$, **b** $N = 50$

delay-bandwidth product of 200 packets on the bottleneck link. This scenario is presented later in Fig. 12. Therefore, we have performed the simulations for four cases $N = 25$, 50, 75, and 100 FTP sources that respectively correspond to light traffic, moderate traffic, high traffic, and very high traffic with respect to the setting shown in Fig. 2. In our scenario, the most significant delay is the queuing delay which depends on the transmission delay. Queuing delay is proportional to the product of the number of packets transmitted and transmission delay. To have a lower queuing delay we consider high bandwidth of the bottleneck link which results in low transmission delay for a given fixed size of packets. In order to demonstrate the efficacy of the proposed algorithm, we examine the performance measures viz. throughput, link-utilization, queue size, and queuing delay through ns2 simulations.

4.1.1 Scenario-1: number of FTP sources $N = 25$

For $N = 25$ FTP sources, throughput is found to increase for AQMRD when all the 25 FTP sources are superimposed but Adaptive RED is capable of achieving more throughput at low traffic loads. Fig. 6 shows a comparative study of throughput for different AQM algorithms. This has been achieved as a result of the adaptation of max_p in the Adaptive RED scheme. Our scheme performs better than Adaptive RED for moderate traffic load as discussed in scenario-2. PI outperforms the AQM schemes because it is able to dynamically control the queue for this network scenario. This is achieved due to the fact that a sufficient buffer size is set and PI is found to be very sensitive towards buffer size.

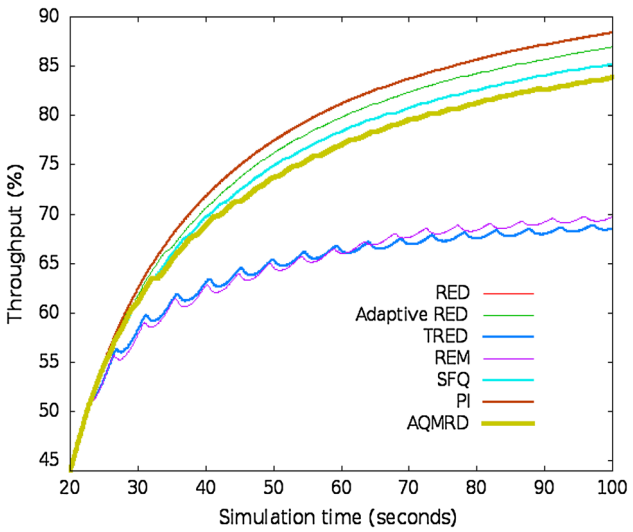

Fig. 6 Comparative study of throughputs for $N = 25$ (Color figure online)

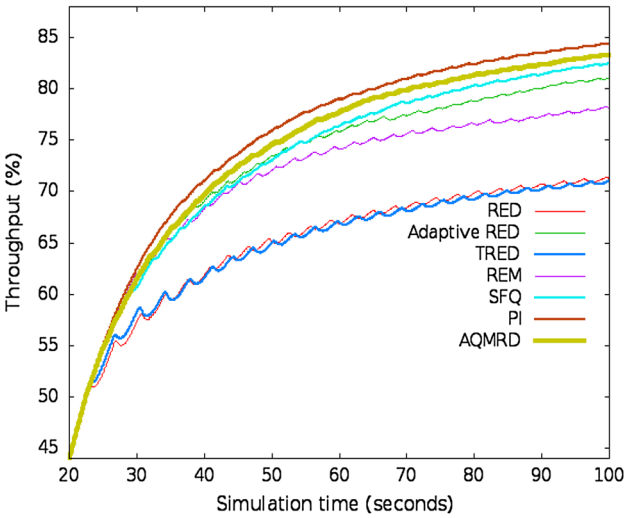

Fig. 7 Comparative study of throughputs for $N = 50$ (Color figure online)

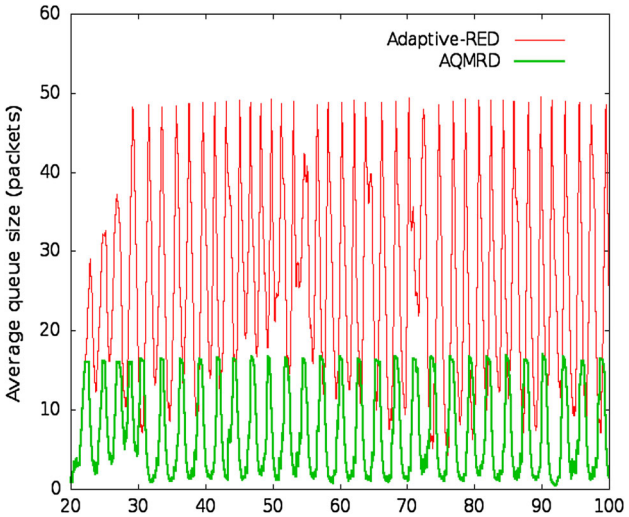

Fig. 8 Comparative study of avg for $N = 50$: Adaptive-RED vs AQMRD (Color figure online)

4.1.2 Scenario-2: number of FTP sources $N = 50$

Next, we increase the number of FTP sources to 50 in order to simulate a network scenario for moderate traffic load. We also compare the throughput with Adaptive RED in Fig. 7 which indicates that Adaptive RED fails when traffic load increases and our scheme performs better than the Adaptive RED scheme. In this setting as well for the earlier mentioned reasons, PI is seen to exhibit the best results but is closely followed by AQMRD. We compare the average queue size changes over the simulation time with Adaptive-RED, see Fig. 8. The proposed AQMRD achieves better results than the other AQM algorithms except PI in terms of throughput. AQMRD also achieves better stabilization of the queue as compared to Adaptive RED.

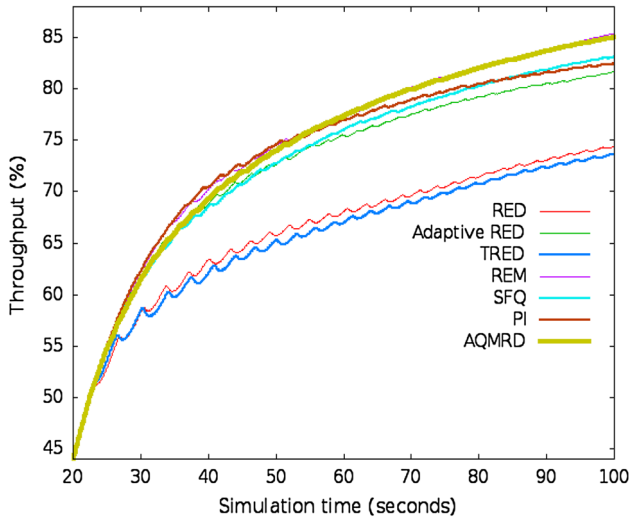

Fig. 9 Comparative study of throughputs for $N = 75$ (Color figure online)

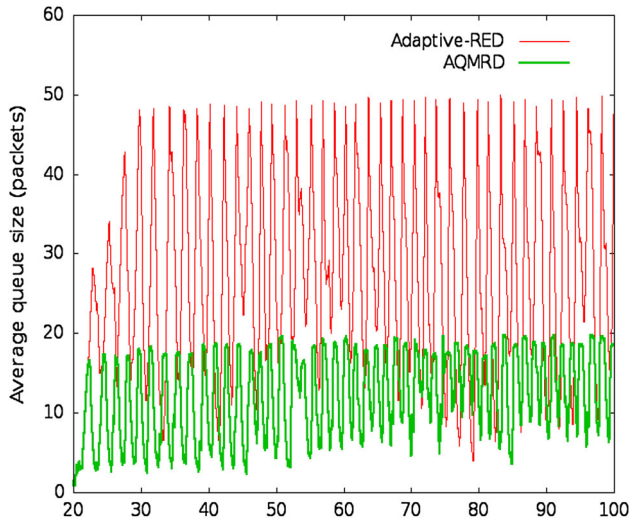

Fig. 10 Comparative study of avg for $N = 75$: adaptive-RED versus AQMRD (Color figure online)

4.1.3 Scenario-3: Number of FTP sources $N = 75$

In order to have a more realistic scenario, we further increase the number of FTP sources to 75. Figure 9 shows the comparative study of throughput for each scheme. We observe that AQMRD shows good results here, better than PI and all the other schemes except REM. For this setting, throughput of REM is very close to our scheme. We also evaluate the average queue size for $N = 75$ and observe that AQMRD is better than Adaptive RED. Figure 10 compares the average queue size stabilization between Adaptive-RED and our scheme. Our scheme AQMRD gets higher stabilization for queue size than Adaptive-RED which results in lower queuing delay for our scheme. It may be noted that whereas Adaptive RED has wide fluctuations in average queue length, our algorithm AQMRD controls these significantly (see Figs. 8, 10). Our scheme is designed so as to be able to control the traffic whether it is with low or high loads. Simulation results achieved in Fig. 9 show that our scheme outperforms all other schemes except REM at high traffic loads. However, the difference in performance between AQMRD and REM is marginal here.

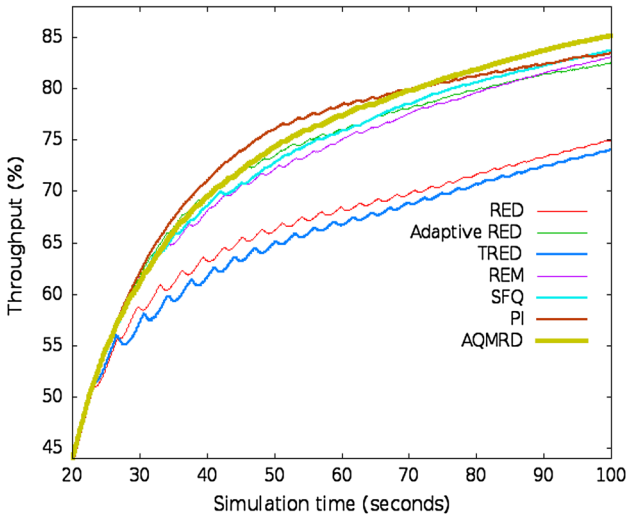

Fig. 11 Comparative study of throughputs for $N = 100$ (Color figure online)

4.1.4 Scenario-4: Number of FTP sources $N = 100$

In order to simulate very heavy traffic, we further increase the number of FTP sources to 100. From Fig. 11, we observe that our scheme outperforms all the other AQM algorithms—RED, MRED, Adaptive-RED, TRED, REM, SFQ, and PI. Scenario-3 and scenario-4 show that our scheme is able to control the congestion better than all other schemes at high as well as very high traffic loads.

4.2 Effect of load

Next, we show the results of several experiments by using the same network parameters as before. Here, we vary the number of FTP sources from 12 to 100 and compare the throughputs in Fig. 12 for eight different levels of traffic load. The purpose of these simulations on ns-2 is to evaluate the average throughput as a function of N. The AQMRD gateway shows improvement in throughput for each simulation and in fact shows the best results when the number of FTP sources goes beyond 54. These results show that the proposed scheme is capable of yielding good performance, regardless of the level of traffic. PI is capable of achieving the best average throughput at light traffic load but its performance degrades at high traffic loads. In comparison, our

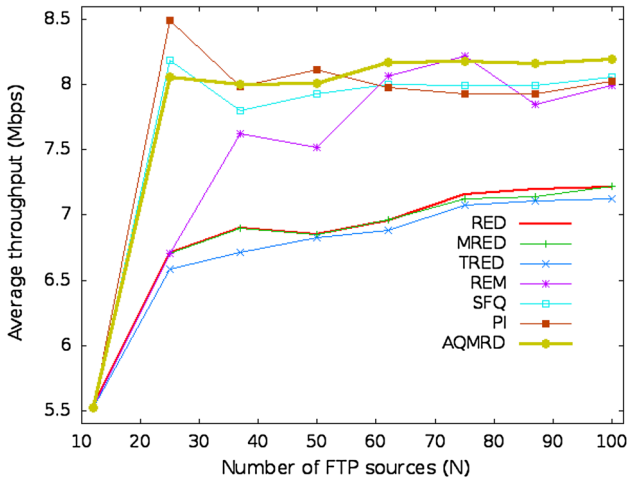

Fig. 12 Comparative study of the average throughput versus N (Color figure online)

scheme is competitive against PI and in fact, consistently outperforms PI at traffic loads beyond 54 sources. The REM algorithm shows good results in a narrow change and is marginally better than AQMRD for $N = 75$ but does not compare favourably at other load levels. Unlike the other algorithms, our scheme exercises better control over the network traffic.

4.3 Effect of max_{th}

Next, we consider the same network parameters as before and evaluate the link utilization or relative throughput (i.e., throughput/bandwidth) for three different levels of max_{th}. We compare the relative throughput of our scheme with that of the recently proposed scheme TRED. It is remarkable that the relative throughput for our scheme approximately increases between 12 and 16 % over TRED for three differ-

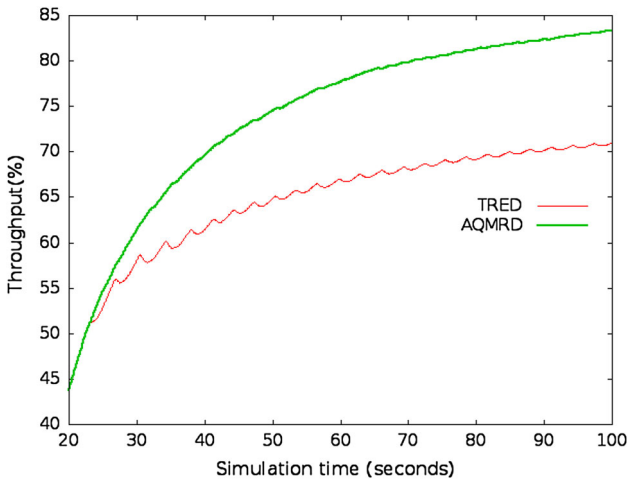

Fig. 13 Throughput comparison of AQMRD versus TRED, $max_{th} = 48$ (Color figure online)

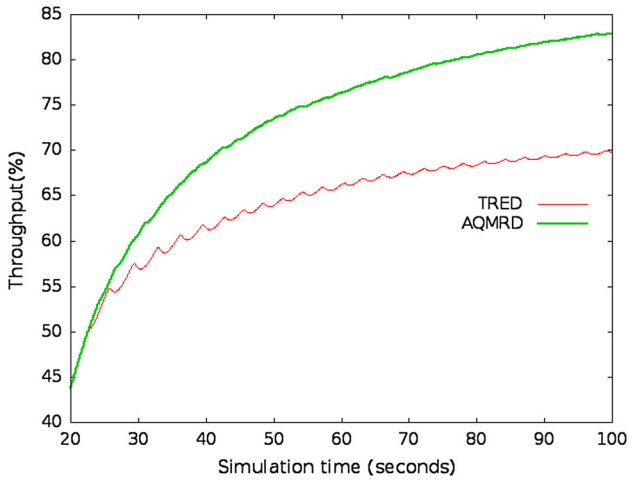

Fig. 14 Throughput comparison of AQMRD versus TRED, $max_{th} = 30$ (Color figure online)

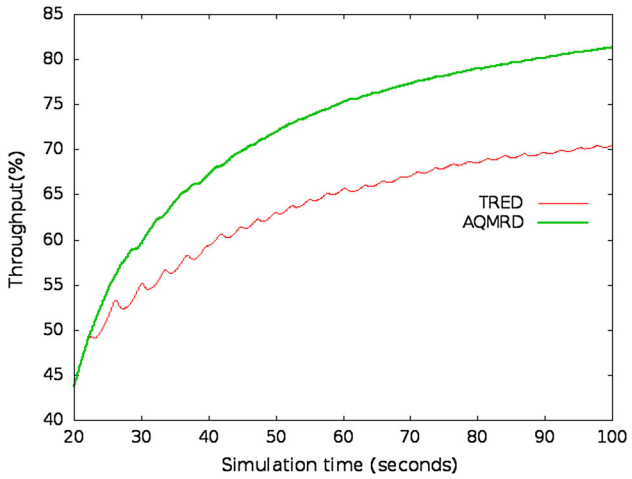

Fig. 15 Throughput comparison of AQMRD versus TRED, $max_{th} = 18$ (Color figure online)

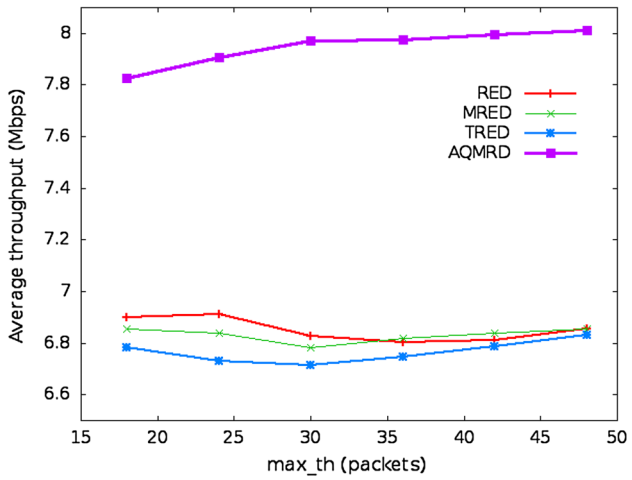

Fig. 16 Comparative study of the average throughput versus max_{th}: RED, MRED, TRED and AQMRD (Color figure online)

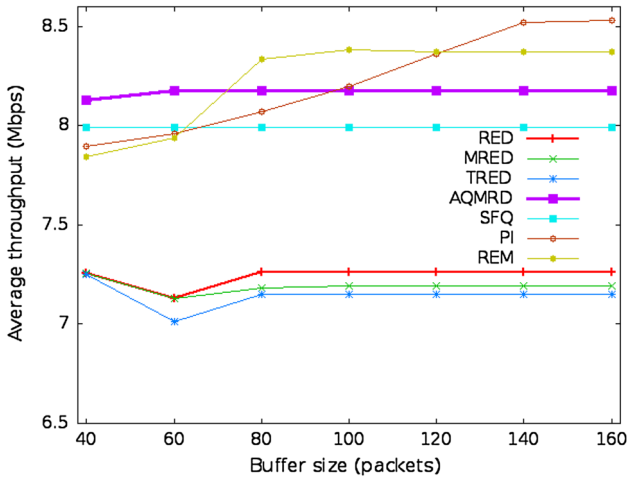

Fig. 17 Comparative study of the average throughput versus buffer size: RED, MRED, TRED, SFQ, REM, PI and AQMRD (Color figure online)

ent levels of max_{th}, see Figs. 13, 14 and 15. We have not shown the relative throughput for RED and MRED because there is only a marginal change in throughput in the schemes over TRED. In Fig. 16 we compare the average throughput of AQMRD with RED, MRED and TRED. The average throughput is computed as the ratio of total number of bytes sent to the total simulation time. The average throughput in each case is evaluated for six different simulations as max_{th} varies from 18 to 48 in equal intervals. All the parameters and the variables other than max_{th} are the same as before. The performance comparisons of the average throughput versus max_{th} of our algorithm with RED, MRED, and TRED are given in Fig. 16. We evaluate the average and relative throughput for different levels of max_{th} and observe from Figs. 13, 14, 15 and 16 that AQMRD performs better than RED, MRED, and TRED, in the relative throughput metric as well.

4.4 Effect of buffer size

In order to check the sensitivity of the buffer size in high traffic load, we performed simulations with buffer sizes varying between 40 and 160 and fixing max_{th} to 48 for each simulation. We consider the simulation for high trafic load with $N = 75$ FTP sources. We have performed seven simulations to carry out the results of throughput as a function of the buffer size. Figure 17 indicates that our proposed scheme uniformly achieves more than 12 % higher throughput than the other algorithms except the two schemes REM and PI. The reason for enhanced throughput in PI and REM is due to the availability of buffer size which can accommodate large queues. This advantage of large buffer size, however, is not available in RED and AQMRD as the queue size is limited to the interval $[min_{th}, max_{th}]$. It is to be underlined that one has to pay the price for increased buffer size. Simulation results also show that PI and REM are sensitive to the buffer size on account of a higher range of queue size. AQMRD outperforms all AQM schemes at low buffer size due to the introduction of the rate of change of average queue size parameter. In fact, our scheme is more robust and shows the least sensitivity or variation in performance as a function of the buffer size.

4.5 Queuing delay

Queuing delay plays an important role under high traffic conditions. We have performed simulation experiments to compare the instantaneous queuing delay and throughput for high traffic with $N = 75$ and $N = 100$ FTP sources. In Figs. 18 and 19, we show plots of the instantaneous queueing delay as a function of simulation time. It can be seen that TRED achieves a low delay with approximately same throughput as RED under high load. Our scheme reduces the queuing delay significantly over TRED even while resulting in higher throughput. Figure 18 shows that for $N = 75$, AQMRD exhibits smaller queuing delay in relation to RED, Adaptive-RED, and TRED. The AQMRD gateway achieves significantly lower queuing delay and higher throughput than each of the schemes RED, MRED, TRED, and Adaptive-RED. From Fig. 19, we also observe that there is much improvement in queuing delay when using AQMRD. The variability in the queueing delay in all the plots shown (with the various AQM schemes) arises because of the stochasticity in the traffic patterns. It is important to note that the aforementioned variability is the least in AQMRD in comparison with all the other AQM schemes. We are able to achieve this through the efficient use of the rate of change of average queue size metric in our algorithm. Our proposed scheme achieves better stabilization of the queuing delay metric with lower delay values achieved as compared to the other schemes which is an important gain for AQMRD. We also obtain good throughput as shown in Fig. 17 which conclusively demonstrates that AQMRD achieves low delay without affecting the goal of achieving high throughput.

4.6 Mean values of performance measures

Expected values of performance measures such as queuing delay, average queue size, instantaneous queue size, and average loss-ratio are given in Tables 1, 3, 5, and 7, respectively. PI is unable to attain the acceptable queuing delay as its expected queuing delay increases by 300 % drastically with respect to RED. On the other hand, it can be seen that the AQMRD gateway achieves upto 17.74 % reduc-

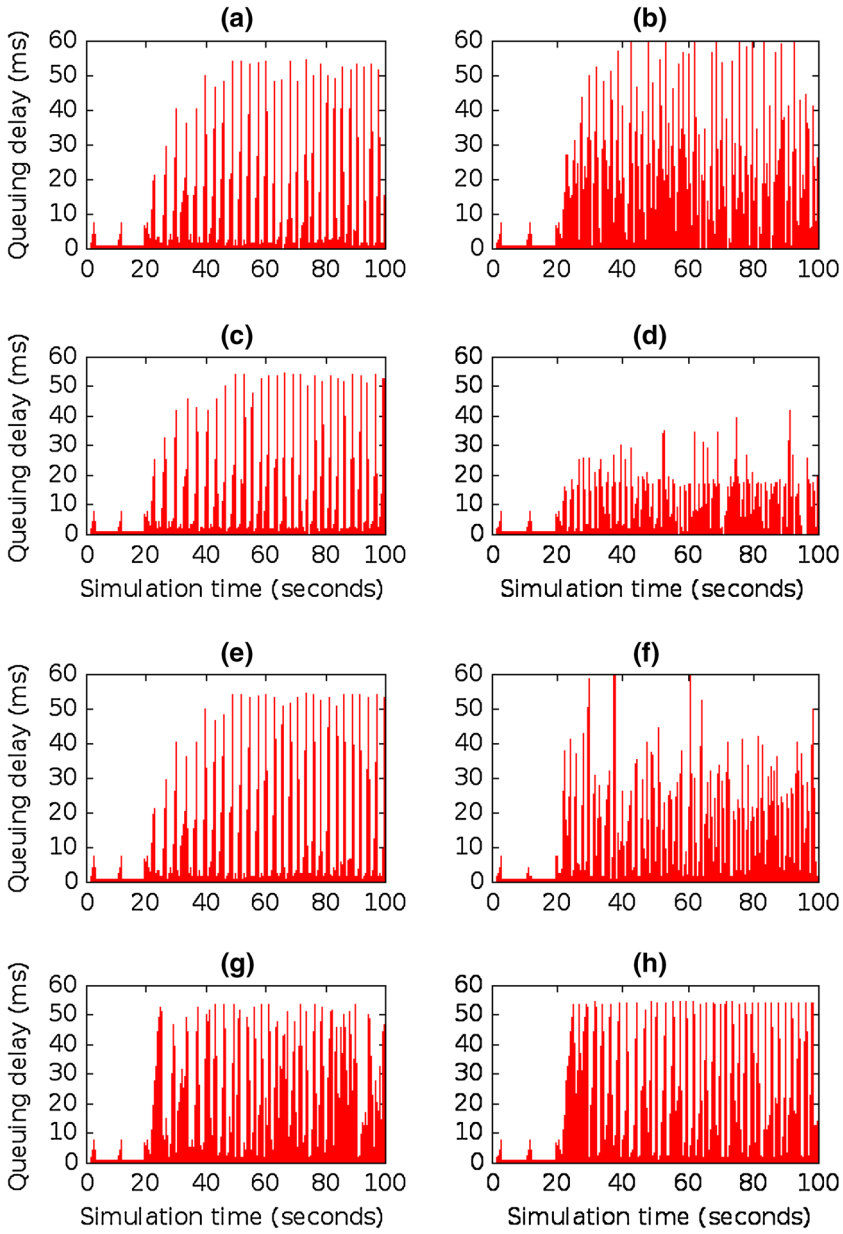

Fig. 18 Comparisons of queuing delay for $N = 75$. **a** RED, **b** Adaptive-RED, **c** TRED, **d** AQMRD, **e** MRED, **f** SFQ, **g** REM, **h** PI

tion in expected queuing delay. Tables 2, 4, and 6 show the percentage reduction in queuing delay, expected value of the average queue size, and expected value of instantaneous queue size each with respect to the RED scheme. Table 8 shows the percentage increase in loss-ratio for each scheme with respect to RED. This reduction in queuing delay is on account of lower expected value of average and instantaneous queue sizes observed in AQMRD. The trade-off between queuing delay and utilization or throughput is also pointed out by Hollot et al. [15] as increase in utilization also results in an increase in the queuing delay. Further, in support of these findings, Feng et al. (see Sect. 2, C of [14]) point out that it may not be possible to achieve both high link utilization/throughput and low queuing delay simultaneously. However, it is worth noting that AQMRD has the advantage of improving both throughput and queuing delay albeit with a higher packet loss rate. It should be emphasized that our AQMRD algorithm significantly improves performance in both the aforementioned QoS metrics.

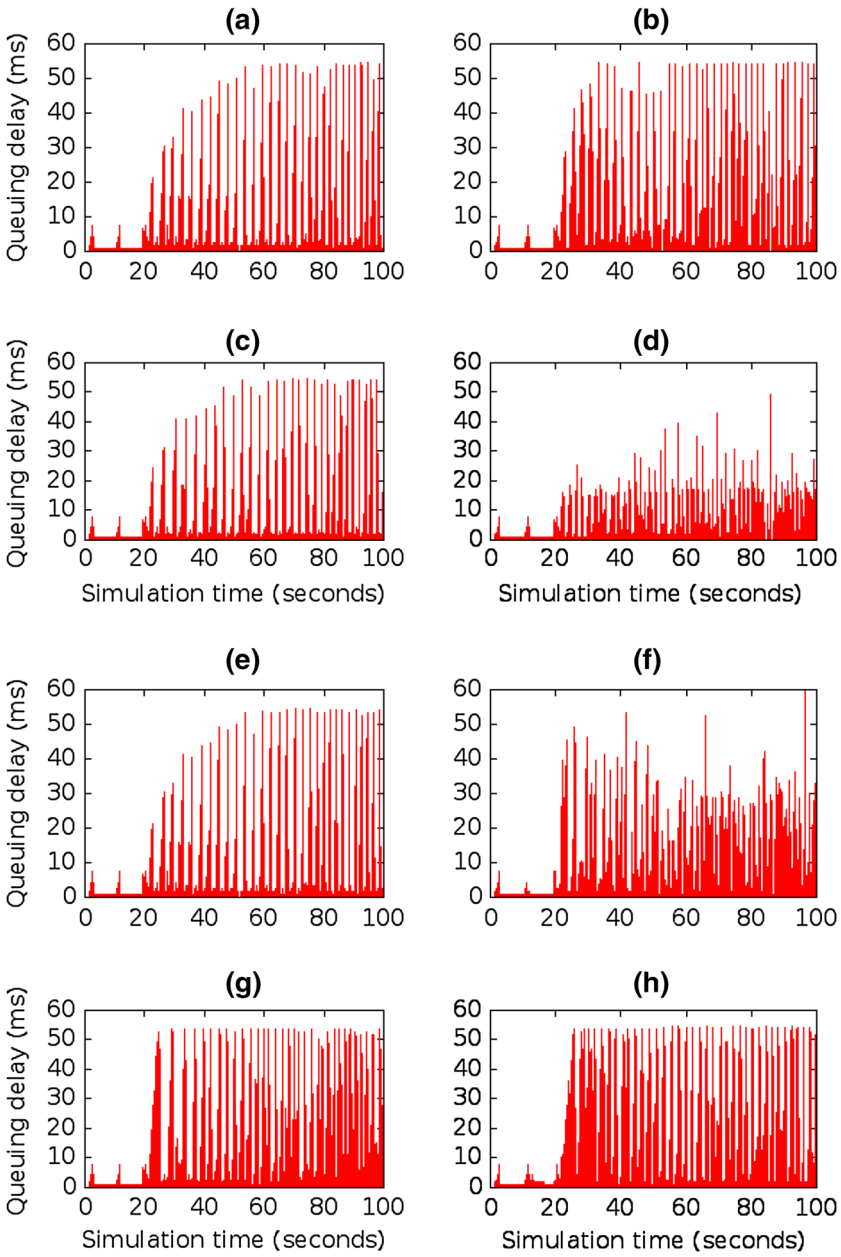

Fig. 19 Comparisons of queuing delay for $N = 100$. **a** RED, **b** Adaptive-RED, **c** TRED, **d** AQMRD, **e** MRED, **f** SFQ, **g** REM, **h** PI

Table 1 Comparative performance results: queuing delay

AQM algorithms	E[queuing delay] (ms)			
	$N = 25$	$N = 50$	$N = 75$	$N = 100$
RED	0.03071	0.02671	0.02656	0.02571
MRED	0.03071	0.02672	0.02807	0.02576
Adaptive-RED	0.06678	0.04092	0.03209	0.03208
SFQ	0.03651	0.02206	0.02086	0.02085
REM	0.03071	0.03802	0.04191	0.03425
PI	0.09313	0.05348	0.04498	0.06083
TRED	0.03437	0.02958	0.02642	0.02690
AQMRD	0.02871	0.02413	0.02259	0.02115

Table 2 Percentage reduction in expected queuing delay with respect to RED

AQM algorithms	Number of FTP sources (N)			
	$N = 25$ (%)	$N = 50$ (%)	$N = 75$ (%)	$N = 100$ (%)
MRED	0	−0.04	−5.69	−0.19
Adaptive-RED	−117.45	−53.20	−20.82	−0.19
SFQ	−18.89	+17.41	+21.46	+18.90
REM	0	−42.34	+57.79	−33.22
PI	−300.25	−100.22	−69.35	−136.60
TRED	−11.91	−10.75	+0.53	−4.63
AQMRD	+6.51	+9.66	+14.95	+17.74

Table 3 Comparative performance results: time average of average queue size

AQM algorithms	E[average queue size] (packets)			
	$N = 25$	$N = 50$	$N = 75$	$N = 100$
RED	6.41	11.51	14.02	14.31
MRED	6.41	11.60	14.18	14.51
Adaptive-RED	21.12	23.15	24.67	21.01
TRED	6.97	11.54	14.01	14.80
AQMRD	7.06	7.09	11.30	11.53

Table 4 Percentage reduction in the expected average queue size with respect to RED

AQM algorithms	Number of FTP sources (N)			
	$N = 25$ (%)	$N = 50$ (%)	$N = 75$ (%)	$N = 100$ (%)
MRED	0	−0.78	−1.14	−1.40
Adaptive-RED	−229.49	−101.13	−75.96	−46.82
TRED	−8.74	−0.26	+0.071	−3.42
AQMRD	−10.14	+38.40	+19.40	+19.43

Table 5 Comparative performance results: expected instantaneous queue size

AQM algorithms	E[instantaneous queue size] (packets)			
	$N = 25$	$N = 50$	$N = 75$	$N = 100$
RED	6.60	13.23	17.83	19.34
MRED	6.60	13.23	18.94	19.65
Adaptive-RED	19.11	26.32	28.67	28.01
TRED	7.14	13.11	17.76	20.36
AQMRD	7.91	13.39	15.27	15.56

Table 6 Percentage reduction in the expected instantaneous queue size with respect to RED

AQM Algorithms	Number of FTP sources (N)			
	$N = 25$ (%)	$N = 50$ (%)	$N = 75$ (%)	$N = 100$ (%)
MRED	0	0	−6.23	−1.60
Adaptive-RED	−189.55	−98.94	−60.80	−44.83
TRED	−8.18	+0.91	+0.40	−5.27
AQMRD	−19.85	−1.21	+14.36	+19.55

Table 7 Comparative performance results: average loss ratio

AQM algorithms	Average loss-ratio (%)			
	$N = 25$	$N = 50$	$N = 75$	$N = 100$
RED	0.578	1.771	2.690	3.067
MRED	0.578	1.771	2.911	3.012
Adaptive-RED	0.200	1.94	2.833	3.078
TRED	0.666	1.819	2.910	3.314
REM	0.578	1.262	1.819	2.242
SFQ	0.374	1.845	2.790	3.077
PI	0.199	1.249	1.908	2.177
AQMRD	0.638	2.771	4.086	4.490

Table 8 Percentage increase in average loss-ratio over RED

AQM algorithms	Number of FTP sources (N)			
	$N = 25$ (%)	$N = 50$ (%)	$N = 75$ (%)	$N = 100$ (%)
MRED	0	0	+8.22	−1.79
Adaptive-RED	−65.40	+9.54	+5.32	+0.36
TRED	+15.19	+2.71	+8.18	+8.05
REM	0	−28.74	−32.78	−26.89
SFQ	−35.29	+4.18	+3.72	+0.33
PI	−65.57	−29.48	−29.07	−29.02
AQMRD	+10.38	+56.47	+51.90	+46.40

5 Conclusions

We presented in this paper a new AQMRD algorithm that incorporates both average queue size and its rate of change, and is found to achieve significantly better performance than RED as well as its variants in the presence of non-stationary heavy traffic. An important aspect of the proposed approach is that it incorporates the rate of change in queue size as an additional parameter. AQMRD prevents the frequent crossing of maximum threshold by the average queue size and rapidly responds to congestion before the overflow of packets occurs. As noted by Feng et al. [14], the principal goal of the congestion controlling algorithm is to maintain low level of the queue size so as to keep low delay. Based on extensive numerical experiments, we have found that AQMRD has outperformed the existing well known AQM algorithms viz., RED, MRED, TRED and Adaptive RED, respectively, as can be seen in Figs. 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18 and 19. Our scheme AQMRD also outperforms the PI and REM at low buffer sizes as also high traffic loads. The proposed AQMRD algorithm has the advantage of reducing the delay and queue size in comparison to the existing RED algorithm but PI is seen to have the worst queuing delay. An important finding here is that the AQMRD gateway reduces the expected value of average queue size by 38.4 % for 50 FTP sources. We may point out that the inclusion of the new parameters corresponding to the rate of change of average queue size as well as the mid_{th} threshold level have played a significant role in enhancing the performance of the algorithm. An area of future enquiry would be to optimize the values

of mid_{th} and the other parameters max_{th}, min_{th}, w_q, max_p etc. using a stochastic optimization approach.

References

1. Floyd, S., & Jacobson, V. (1993). Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, *1*(4), 397–413.
2. Athuraliya, S., Li, V. H., Low, S. H., & Elissa, Q. Y. (2002). REM: Active queue management. *IEEE Network*, *15*(3), 48–53.
3. Feng, W., Shin, K. G., Kandlur, D. D., & Saha, D. (2002). The BLUE active queue management algorithms. *IEEE/ACM Transactions on Networking*, *10*(4), 513–528.
4. Meckenney, P. E. (1990). Stochastic fair queuing. In *Proceedings of IEEE INFOCOM* (pp. 733–740, Vol. 2).
5. Floyd, S., Gummadi, R., & Shenker, S. (2001). Adaptive RED: An algorithm for increasing the robustness of RED's active queue management. Berkeley, CA. http://www.icir.org/floyd/papers.html.
6. May, M., Bolot, J., Diot, C., & Lyles, B. (1999). Reasons not to deploy RED. In *Proceedings of IWQoS* (pp. 260–262)
7. Wang, H., & Shin, K.G. (1999). Refined design of random early detection gateways. In *Proceedings of IEEE GLOBECOM* (pp. 769–775).
8. Feng, G., Agarwal, A. K., Jayaraman, A., & Siew, C. K. (2004). Modified RED gateways under bursty traffic. *IEEE Communications Letters*, *8*(5), 323–325.
9. Low, S. H., Paganini, F., Wang, J., & Doyle, J. C. (2003). Linear stability of TCP/RED and a scalable control. *Computer Networks*, *43*(5), 633–647.
10. Tan, L., Zhang, W., Peng, G., & Chen, G. (2006). Stability of TCP/RED systems in AQM routers. *IEEE Transactions on Automatic Control*, *51*(8), 1393–1398.
11. Woo, S., & Kim, K. (2010). Tight upper bound for stability of TCP/RED systems in AQM routers. *IEEE Communications Letters*, *14*(7), 682–684.
12. Bhatnagar, S., & Patro, R. K. (2009). A proof of convergence of the B-RED and P-RED algorithms for random early detection. *IEEE Communications Letters*, *13*(10), 809–811.
13. Adams, R. (2013). Active queue management: a survey. *IEEE Communations Surveys & Tutorials*, *15*(3), 1425–1476.
14. Feng, C. W. , Huang, L. F., Xu, C., & Chang, Y. C. (2015). Congestion control scheme performance analysis based on nonlinear RED, *Journal of IEEE Systems*, *PP*(99), (1–8). [Early Access Article].
15. Hollot, C.V., Misra, V., Towsley, D., & Gong, W. (2001). On designing improved controllers for AQM routers supporting TCP flows. In *Proceedings of IEEE INFOCOM* (pp. 1726–1734).
16. Hollot, C. V., Misra, V., Towsley, D., & Gong, W. (2002). Analysis and design of controllers for AQM routers supporting TCP flows. *IEEE Transactions on Automatic Control*, *47*(6), 945–959.
17. Wu, Y., Min, G., & Yang, L. T. (2013). A network and device aware QoS approach for cloud-based mobile streaming. *IEEE Transactions on Multimedia*, *15*(4), 747–757.
18. Wang, J., Rong, L., & Liu, Y. (2008). A robust proportional controller for AQM based on optimized second-order system model. *Computer Communications*, *31*(10), 2468–2477.
19. Chavan, K., Kumar, R. G., Belur, M. N., & Karandikar, A. (2011). Robust active queue management for wireless neyworks. *EEE Transactions on Control Systems Technology*, *19*(6), 1630–1638.
20. Tse, D., & Viswanath, P. (2005). *Fundamentals of wireless communication*. Cambridge: Cambridge University Press.
21. Wu, Y., Min, G., & Yang, L. T. (2013). Performance analysis of hybrid wireless networks under bursty and correlated traffic. *EEE Transactions on Vehicular Technology*, *62*(1), 449–454.

Karmeshu has been working as Professor in School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, India, since 1986. Prior to joining JNU, he taught at the University of Delhi (India), University of Waterloo (Canada), UAM, and UNAM (Mexico). After obtaining his Ph.D. in 1976 in the area of Diffusion of Brownian Particles Through Generalized Langevin Equation, he joined the University of Stuttgart, Germany, as a visiting scientist. In 1993, Dr. Karmeshu received the Shanti Swarup Bhatnagar Award, India's most prestigious award in the field of science and engineering, for his contributions to mathematical modelling of social and technical systems. He was also awarded the Dr. C. M. Jacob Gold Medal for the year 1990 by the Systems Society of India. Dr. Karmeshu has published about 100 papers in journals. He has been working in the area of modelling and simulation of non-linear stochastic systems covering a wide range of engineering and socio-technical systems. His current research interests lie in the study of mobile and wireless systems, application of entropy framework to communication networks and computational neuroscience.

Sanjeev Patel received his B.Tech. degree in Computer Science and Engineering from Motilal Nehru National Institute of Technology, Allahabad, India in 2005, M.Tech in Computer Science and Technology from Jawaharlal Nehru University, New Delhi, India in 2008. He is currently working as an Assistant Professor in the Department of CSE at Jaypee Institute of Information Technology Noida. He is also a research scholar at School of Computer and System Sciences, Jawaharlal Nehru University, New Delhi, India and is working in the area of performance modeling of flow and congestion control.

Shalabh Bhatnagar received a Bachelors in Physics (Hons) from the University of Delhi in 1988. He received his Masters and Ph.D. degrees in Electrical Engineering from the Indian Institute of Science, Bangalore in 1992 and 1997, respectively. He was a Research Associate at the Institute for Systems Research, University of Maryland, College Park, during 1997 to 2000 and a Divisional Postdoctoral Fellow at the Free University, Amsterdam, during 2000 to 2001. He is currently working as a Professor at the Department of Computer Science and Automation at the Indian Institute of Science, Bangalore. He has

also held visiting positions at the Indian Institute of Technology, Delhi and the University of Alberta, Canada. Dr.Bhatnagar's interests are in simulation optimization, stochastic control and reinforcement learning. He has authored or co-authored more than 115 research articles in various journals and conferences. He is the Principal at author of a book with title 'Stochastic Recursive Algorithms for Optimization: Simultaneous Perturbation Methods', published by Springer in 2013. He is a Fellow of the Indian National Academy of Engineering and a Fellow of the Institution of Electronics and Telecommunication Engineers.