Journal of Network and Computer Applications 35 (2012) 857–864

Contents lists available at SciVerse ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

CARED: Cautious Adaptive RED gateways for TCP/IP networks

Mohit P. Tahiliani [a,*], K.C. Shet [a], T.G. Basavaraju [b]

[a] Department of Computer Science and Engineering, N.I.T.K., Surathkal, Mangalore 575025, Karnataka, India
[b] Department of Computer Science and Engineering, G.E.C., Ramanagara 571511, Karnataka, India

ARTICLE INFO

Article history:
Received 2 June 2011
Received in revised form
18 October 2011
Accepted 5 December 2011
Available online 14 December 2011

Keywords:
Active queue management
Packet drop rate
Throughput

ABSTRACT

Random Early Detection (RED) is a widely deployed active queue management algorithm that improves the overall performance of the network in terms of throughput and delay. The effectiveness of RED algorithm, however, highly depends on appropriate setting of its parameters. Moreover, the performance of RED is quite sensitive to abrupt changes in the traffic load. In this paper, we propose a Cautious Adaptive Random Early Detection (CARED) algorithm that dynamically varies maximum drop probability based on the *level of traffic load* to improve the overall performance of the network. Based on extensive simulations conducted using Network Simulator-2 (*ns*-2), we show that CARED algorithm reduces the packet drop rate and achieves high throughput as compared to RED, Adaptive RED and Refined Adaptive RED. Unlike other RED based algorithms, CARED algorithm does not introduce new parameters to achieve performance gain and hence can be deployed without any additional complexity.

1. Introduction

Tremendous growth in the number of internet users and a high demand for continuous network connectivity have led to an exponential increase in the internet traffic, making it complicated to handle network congestion. TCP congestion control mechanisms are widely deployed in well known operating systems and are extensively used by a variety of internet applications such as electronic mail, file transfer, etc. Since most of these mechanisms consider network as a *black box*, they are limited to *packet drops* as the only indication of congestion. The major limitation of such mechanisms is that they are not well suited for applications such as telnet, web browsing, etc. which are sensitive to packet drops. Moreover, traditional drop-tail gateways do not provide an early congestion notification. This leads to global synchronization, a phenomenon in which all senders sharing the bottleneck gateway reduce their sending rate at the same time, thereby under-utilizing the network resources.

Recently, another drawback of drop-tail gateways known as *Bufferbloat* (Gettys, 2011) has drawn attention of several researchers. Since memory costs have reduced in the recent past, modern routers are designed with extremely large buffers. TCP variants implemented in present operating systems do not reduce the sending rate unless a packet drop is encountered. These reactive mechanisms fill buffers of any capacity before reducing to achieve their fair share of bandwidth. Since the packet drop occurs only when these large buffers overflow, queuing delay experienced by each packet increases drastically, thereby degrading the Quality of Service for delay sensitive applications such as DNS queries, Voice over IP (VoIP) and other multimedia applications.

Congestion avoidance mechanisms differ from congestion control mechanisms, since former are proactive while latter are reactive. Active Queue Management (AQM) mechanisms at gateways have been extensively studied in the recent past to *avoid congestion*. Moreover, AQM mechanisms also seem to be promising solution to avoid *Bufferbloat* problem as well. Random Early Detection (RED) gateways (Floyd and Jacobson, 1993) overcome the drawbacks of drop-tail gateways by avoiding global synchronization and offer fairness among several competing end hosts. However, it has been shown that the effectiveness of RED largely depends on appropriately setting atleast four parameters, namely: minimum threshold (min_{th}), maximum threshold (max_{th}), queue weight factor (w_q) for exponential weighted moving average and maximum drop probability (max_p) (Feng et al., 1999, 2001; Floyd et al., 2001; Kunniyur and Srikant, 2001; Lakshman et al., 1999). Optimal values for these parameters differ for different scenarios and are dependent on several other factors such as number of flows passing through same bottleneck gateway (Feng et al., 1999, 2001), packet size (Misra et al., 2000), etc. Table 1 presents the values of above mentioned parameters used in Cisco 12000 Series routers that implement a modified RED called Weighted RED (WRED) (Weighted Random Early Detection, 2009). C is the capacity of the link in packets where mean packet size is 1500 bytes.

Adaptive RED (ARED) algorithm (Floyd et al., 2001) addresses the parameter sensitivity of RED by dynamically varying max_p and automatically setting min_{th}, max_{th} and w_q parameters. ARED requires setting of only one parameter—*target queuing delay*, defined as the maximum amount of time a packet is delayed in the queue. In this

* Corresponding author. Tel.: +91 2637251473.
E-mail address: tahiliani.nitk@gmail.com (M.P. Tahiliani).

1084-8045/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2011.12.003

Table 1
WRED parameter setting in Cisco 12 000 Series Router [Weighted Random Early Detection (2009)].

Link speed	C	min_{th}	max_{th}	w_q	max_p
DS3	3666	110	367	9	1
OC3	12 917	388	1292	10	1
OC12	51 666	1550	5167	12	1

paper we demonstrate that ARED adapts max_p conservatively which leads to degradation of throughput whereas Refined Adaptive RED (Re-ARED) (Kim and Lee, 2006) adapts max_p aggressively which leads to more packet drops.

Unlike ARED and Re-ARED that adapt max_p conservatively and aggressively respectively, we propose a Cautious Adaptive Random Early Detection (CARED) algorithm that adapts max_p either aggressively or conservatively depending on the *level of traffic load*. Simulations carried out on *ns*-2 (UCN/LBL/VINT, 2011) demonstrate that by varying max_p with respect to the *level of traffic load*, CARED algorithm improves the overall performance of the network by reducing packet drop rate and achieving high throughput as compared to RED, ARED and Re-ARED.

The remainder of the paper is organized as follows: Section 2 describes the related work that aims to address the parameter sensitivity of RED. Section 3 presents a comparative study of ARED and Re-ARED. Section 4 provides details on the proposed modifications of Re-ARED and the design of CARED algorithm. Section 5 demonstrates the results and Section 6 concludes the paper.

2. Literature review and related work

The parameter sensitivity of RED has been addressed by several researchers and as a result, RED has been extended and enhanced by adopting many different approaches. The basic mechanism of RED, however, still remains same.

On arrival of each packet, RED gateways calculate average queue size (*avg*) using Exponential Weighted Moving Average (EWMA). If *avg* is less than min_{th}, the packet is enqueued. If *avg* is more than max_{th}, the packet is dropped. However, if *avg* is between min_{th} and max_{th}, the packet is dropped randomly with a certain probability. The following equations show *avg* and packet drop probability (p_d) calculation of RED respectively:

$$avg = ((1-w_q) \times oldavg) + (w_q \times cur\_q) \quad (1)$$

where *oldavg* is the average queue size during previous packet arrival; *cur_q* is the current queue size

$$p_d = \begin{cases} 0 & avg < min_{th} \\ \dfrac{avg - min_{th}}{max_{th} - min_{th}} \times max_p & min_{th} \le avg < max_{th} \\ 1 & avg \ge max_{th} \end{cases} \quad (2)$$

The probability with which a packet is dropped is a linear function of *avg*. Hence when *avg* varies from min_{th} to max_{th}, the drop probability varies from 0 to maximum drop probability max_p. If *avg* increases above max_{th}, drop probability becomes 1 i.e. all incoming packets are dropped. It is observed that sharply increasing the drop probability to 1 when *avg* crosses max_{th} results in high number of packet drops. Hence, a modified RED known as Gentle RED (GRED) is proposed by Floyd that varies drop probability from max_p to 1 when *avg* varies from max_{th} to twice max_{th} so as to reduce the number of packet drops.

In Feng et al. (1999) authors show that the effectiveness of RED mechanism largely depends on appropriately setting the four parameters. A Self Configuring RED is proposed that varies max_p parameter based on the queue length dynamics. Moreover, the max_p parameter is varied so as to keep the *avg* between min_{th} and max_{th}.

As an extension to Self Configuring RED, an Adaptive RED (ARED) is proposed in Floyd et al. (2001). ARED aims to keep the *avg* in *target range* between min_{th} and max_{th} and hence varies max_p accordingly. Since ARED follows Additive Increase Multiplicative Decrease (AIMD) policy to vary max_p, it reacts conservatively to abrupt changes in the traffic load.

Several other variants of ARED have also been proposed: a RED based algorithm that adaptively varies w_q along with max_p is proposed in Verma et al. (2002). Similar mechanisms, Stabilized ARED (SARED) (Javam and Analoui, 2006) and Self Tuning RED (Chen et al., in press) focus on assigning different queue weights w_q to ARED instead of one fixed queue weight. The major limitation of these approaches is that they introduce several new parameters to achieve performance gain. Setting these additional parameters adds to the complexity.

Refined Adaptive RED (Re-ARED) proposed in Kim and Lee (2006) aims to bring *avg* within its *target range* more *quickly* by varying max_p aggressively. A modified ARED algorithm based on Multiplicative Increase Multiplicative Decrease (MIMD) policy to adapt max_p is proposed in Marquez et al. (2007). However, authors conclude that MIMD policy to adapt max_p yields similar results as AIMD policy.

Another category of RED based AQM mechanisms not only takes *average queue size* into consideration but also considers the *instantaneous queue size* at the gateway to reduce the packet drop rate and improve the overall throughput. Examples of such mechanisms include Modified RED (MRED) (Feng et al., 2004) and Effective RED (ERED) (Abbasov and Korukoglu, 2009). However, appropriately setting thresholds for *instantaneous queue size* is a challenging issue in these mechanisms. Zhou et al. (2006) propose a Nonlinear RED (NLRED) that replaces the linear packet dropping function of RED by a nonlinear quadratic function to improve the effectiveness of RED mechanism. Since these mechanisms are completely based on the basic RED algorithm, the parameter sensitivity of these mechanisms remains same as that of the basic RED.

Several other AQM mechanisms based on RED have been proposed: Double Slope RED (DS-RED) (Zheng and Atiquzzaman, 2000), Dynamic RED (DRED) (Aweya et al., 2001), RED with Preferential Dropping (RED-PD) (Mahajan and Floyd, 2001), Exponential RED (Liu et al., 2005), Loss-ratio based RED (LRED) (Wang et al., 2007), AQM mechanism based on Neural Networks (NN-RED) (Hariri and Sadati, 2007), etc. There are some concerns on the suitability of approaches followed by all these mechanisms since they do not eliminate the parameter sensitivity of RED. Moreover these mechanisms are more complicated to deploy than the basic RED algorithm.

3. Cautious Adaptive Random Early Detection (CARED)

3.1. Motivation

ARED's fixed and conservative approach of adapting max_p leads to degradation of throughput when level of congestion changes abruptly, especially in light and moderate traffic load scenarios. We call the time period during which level of congestion changes abruptly as "*critical time period*". Since ARED adapts max_p conservatively (Floyd et al., 2001), it does not drop sufficient packets so as to keep *avg* within *target range* during *critical time period*. As a result, the overall throughput degrades because *avg* goes out of *target range* for a small amount of time till max_p increases/decreases to the desired value to bring *avg* back within the *target range*.

Re-ARED addresses the drawback of ARED and adapts max_p based on the ratio of the change in the average queue size that

infers changes in the traffic load. This mechanism improves the throughput of the network in light as well as moderate traffic load scenarios. However, when traffic load is high, it does not eliminate the drawbacks of ARED algorithm.

3.2. Comparative study of ARED and Re-ARED

Re-ARED algorithm as proposed in Kim and Lee (2006) is shown in Algorithm 1. A comparative study of ARED and Re-ARED is carried out using *ns*-2. We simulate different scenarios by varying the number of connections and type of traffic in a dumbbell topology as shown in Fig. 1. Bottleneck bandwidth is fixed to 10 Mbps and RTT propagation delay is set to 80 ms. A burst of packets is sent at the bottleneck router in the beginning of the simulation. This results in sharp increase in the average queue length.

We observe the average queue length dynamics while ARED and Re-ARED attempt to bring average queue size back within *target range*. Fig. 2 shows average queue length dynamics of ARED and Re-ARED for one such scenario.

Algorithm 1. Re-ARED algorithm.

every interval seconds:
if $avg < target$ **and** $max_p \geq 0.01$ **then**
  decrease max_p
  $\beta = 1 - \left(0.17 \times \frac{target - avg}{target - min_{th}}\right)$
  $max_p = max_p \times \beta$
end
else if $avg > target$ **and** $max_p \leq 0.5$ **then**
  increase max_p
  $\alpha = 0.25 \times max_p \times \frac{avg - target}{target}$
  $max_p = max_p + \alpha$
end

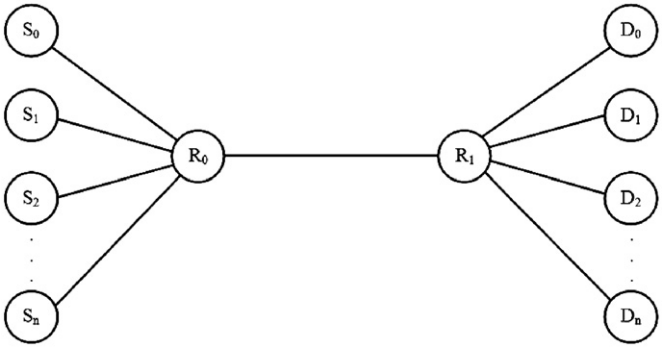

Fig. 1. Dumbbell topology.

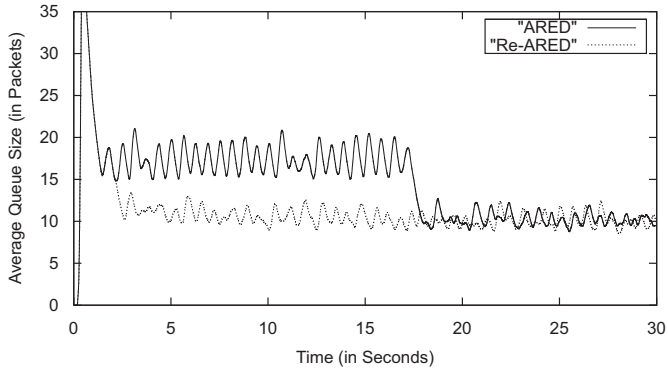

Fig. 2. Average queue length dynamics of ARED and Re-ARED.

Variables:
avg: average queue size;
β : decrease parameter;
α : increase parameter;
Fixed parameters:
interval = 0.5 s;
target = target for average queue size:
$[min_{th} + 0.48 \times (max_{th} - min_{th}), min_{th} + 0.52 \times (max_{th} - min_{th})]$.

Based on simulations results we observe that Re-ARED indeed overcomes the drawback of ARED when the traffic load at gateway is light or moderate. From Fig. 2 it can be observed that ARED algorithm takes longer time to bring *avg* back within the *target* range whereas Re-ARED algorithm takes very short time for the same. Table 2 presents simulation results of the study.

Under light traffic load scenarios, Re-ARED achieves far better throughput than ARED at almost same packet drop rate. Under moderate traffic load scenarios Re-ARED achieves similar throughput as ARED but at the cost of increased packet drop rate. However, the major observation of this study is that under heavy traffic load scenarios, ARED is more robust and hence achieves better throughput than Re-ARED.

Thus, a proper combination of ARED and Re-ARED can improve the overall performance of the network under light, moderate as well as heavy traffic load scenarios. As a result, we propose a Cautious Adaptive RED (CARED) algorithm that adapts max_p conservatively like ARED or aggressively like Re-ARED based on the *level of traffic load* so as to reduce packet drop rate and maximize the throughput.

4. Design of CARED algorithm

Before dwelling into the details of Cautious Adaptive RED (CARED), we attempt to improve the performance of Re-ARED algorithm, especially in moderate and heavy traffic load scenarios by making minor modifications to the basic Re-ARED algorithm.

4.1. Modifications to Re-ARED algorithm

While adapting max_p it should be ensured that a single modification of max_p does not exceed average queue size from *below target* to *above target* or vice versa. This can be achieved by appropriately selecting values for α and β. As a result, based on the *target range*, authors of ARED suggest the following upper bound and lower bound for α and β respectively:

$$\alpha < 0.25 \times max_p \tag{3}$$

$$\beta > 0.83 \tag{4}$$

These bounds ensure that single modification of max_p would not exceed average queue length from *below target* to *above target* or vice versa. Note that the bounds are dependent on *target range*.

Table 2
Packet drop rate and throughput of ARED and Re-ARED.

FTP connections	ARED		Re-ARED	
	Packet drop rate (%)	Throughput (kbps)	Packet drop rate (%)	Throughput (kbps)
5	0.22	9145	0.23	9199
50	10.41	9178	10.54	9178
100	17.84	9108	17.80	9097

As shown in Re-ARED algorithm, bounds on α and β are retained, but *target range* is modified to $[min_{th} + 0.48 \times (max_{th} - min_{th}),\ min_{th} + 0.52 \times (max_{th} - min_{th})]$ instead of $[min_{th} + 0.4 \times (max_{th} - min_{th}),\ min_{th} + 0.6 \times (max_{th} - min_{th})]$ as in ARED algorithm. Since bounds on α and β are based on *target range*, if *target range* changes, even bounds must change. By retaining α and β bounds of ARED, but modifying the *target range*, Re-ARED does not ensure that a single modification of max_p would not exceed average queue length from *below target* to *above target* or vice versa.

Hence we make two modifications to original Re-ARED algorithm and compare the performance of resulting algorithms with the original Re-ARED algorithm.

4.1.1. Re-ARED-M1: with modified target range

In this modification we retain the bounds on α and β but modify the *target range* of original Re-ARED from $[min_{th} + 0.48 \times (max_{th} - min_{th}),\ min_{th} + 0.52 \times (max_{th} - min_{th})]$ to $[min_{th} + 0.4 \times (max_{th} - min_{th}),\ min_{th} + 0.6 \times (max_{th} - min_{th})]$. This ensures that the bounds do not allow average queue length to exceed *below target* to *above target* or vice versa. Rest of the algorithm remains same as original Re-ARED.

4.1.2. Re-ARED-M2: with modified α and β bounds

In this modification we retain the *target range* as specified in Re-ARED algorithm but derive new upper bound and lower bound for α and β respectively to ensure that average queue length does not exceed *below target* to *above target* and vice versa. New bound for α is derived as follows:

From Eq. (2) we have

$$p = max_p \times \left(\frac{avg - min_{th}}{max_{th} - min_{th}}\right) \tag{5}$$

Before adapting max_p

$$avg_1 = min_{th} + \frac{p}{max_p} \times (max_{th} - min_{th}) \tag{6}$$

and after adapting max_p

$$avg_2 = min_{th} + \frac{p}{max_p + \alpha} \times (max_{th} - min_{th}) \tag{7}$$

Subtracting (7) from (6)

$$avg_1 - avg_2 = \frac{\alpha}{max_p + \alpha} \times \frac{p}{max_p} \times (max_{th} - min_{th}) \tag{8}$$

Hence to ensure *avg* does not exceed *above target* to *below target*

$$\frac{\alpha}{max_p + \alpha} < 0.04 \tag{9}$$

$$\Rightarrow \alpha < 0.0412 \times max_p \tag{10}$$

Similarly, new bound for β can be obtained as follows:

Before adapting max_p

$$avg_1 = min_{th} + \frac{p}{max_p} \times (max_{th} - min_{th}) \tag{11}$$

and after adapting max_p

$$avg_2 = min_{th} + \frac{p}{max_p \times \beta} \times (max_{th} - min_{th}) \tag{12}$$

Subtracting (12) from (11)

$$avg_1 - avg_2 = \frac{1-\beta}{\beta} \times \frac{p}{max_p} \times (max_{th} - min_{th}) \tag{13}$$

Table 3
Packet drop rate of Re-ARED, Re-ARED-M1 and Re-ARED-M2.

FTP connections	Re-ARED (%)	Re-ARED-M1 (%)	Re-ARED-M2 (%)
5	0.23	0.23	0.22
50	10.54	10.32	10.34
100	17.80	17.87	17.82

Table 4
Throughput (kbps) of Re-ARED, Re-ARED-M1 and Re-ARED-M2.

FTP connections	Re-ARED	Re-ARED-M1	Re-ARED-M2
5	9199	9187	9162
50	9178	9184	9187
100	9097	9111	9107

Hence to ensure *avg* does not exceed *below target* to *above target*

$$\frac{1-\beta}{\beta} < 0.04 \tag{14}$$

$$\Rightarrow \beta > 0.9615 \tag{15}$$

Tables 3 and 4 show simulation results of Re-ARED-M1 and Re-ARED-M2 as compared to Re-ARED for packet drop rate and throughput respectively. Results demonstrate that modified algorithms achieve higher performance gain as compared to the original Re-ARED algorithm. We observe that though Re-ARED-M2 gives promising results when traffic load is moderate, it gives almost similar results to that of Re-ARED when traffic load is heavy. Moreover, Re-ARED-M2 results in drastic decrease in throughput when traffic load is light. However, Re-ARED-M1 gives slightly better performance than Re-ARED when traffic load is heavy while retaining Re-ARED's performance for light as well as moderate traffic load scenarios. Hence, while designing CARED algorithm, instead of merging ARED and original Re-ARED, we merge ARED and Re-ARED-M1 based on the *level of traffic load*.

4.2. CARED algorithm

CARED algorithm is designed to adapt max_p either conservatively or aggressively based on the *level of traffic load*. We classify the *level of traffic load* into: up and down. If current average queue length (*newavg*) is greater than previous average queue length (*oldavg*), the *level of traffic load* is considered as up since the average queue length is increasing. Similarly if current average queue length (*newavg*) is less than previous average queue length (*oldavg*), the *level of traffic load* is considered as down since the average queue length is decreasing. Based on this notion of *level of traffic load*, the proposed CARED algorithm is as shown in Algorithm 2. Setting other parameters such as min_{th}, max_{th}, w_q and *target queuing delay* in CARED is similar to that of ARED.

Design considerations of CARED algorithm are as follows:

- If *newavg* is below $target_{low}$ and the *level of traffic load* is up, max_p is decreased conservatively as per ARED. Aggressively decreasing max_p as per Re-ARED-M1 would result in further increase in sending rate which may take *newavg* above $target_{up}$.
- If *newavg* is below $target_{low}$ and the *level of traffic load* is down, max_p is decreased aggressively as per Re-ARED-M1. Conservatively decreasing max_p as per ARED in this scenario would lead to under-utilization of queue.
- If *newavg* is above $target_{up}$ and the *level of traffic load* is down, max_p is increased conservatively as per ARED. Aggressively

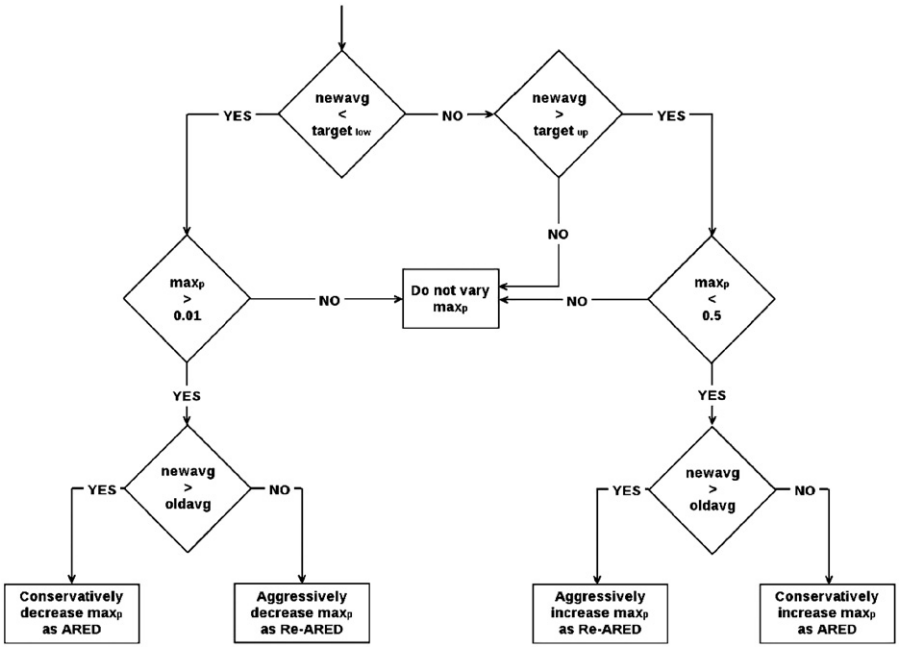

Fig. 3. Flowchart of Cautious Adaptive RED.

increasing max_p as per Re-ARED-M1 in this scenario would drop more packets than required. Since *newavg* is already decreasing and moving towards *target range*, packet drops must be conservative rather than aggressive.

- If *newavg* is above $target_{up}$ and the *level of traffic load* is up, max_p is increased aggressively as per Re-ARED-M1 to avoid *newavg* from crossing max_{th}. Conservatively increasing max_p as per ARED in this scenario would take more time to bring *newavg* back within the *target range* and hence may affect the throughput and also increase queuing delay.

Note that like ARED and Re-ARED, CARED also varies max_p within a range of 1–50%. Fig. 3 demonstrates the design of CARED algorithm:

Variables used in CARED algorithm:
newavg: current average queue size;
oldavg: average queue size during previous interval;
β *: decrease parameter*;
α *: increase parameter*;
Fixed parameters in CARED algorithm:
interval$=0.5$ s;
$\beta = 0.9$ *for ARED*;
$target_{low}$=*lower bound for target*$=min_{th} + 0.4 \times (max_{th} - min_{th})$;
$target_{up}$=*upper bound for target*$=min_{th} + 0.6 \times (max_{th} - min_{th})$.

Algorithm 2. CARED algorithm.

every interval seconds:
if $newavg < target_{low}$ **and** $max_p \geq 0.01$ **then**
  if $newavg > oldavg$ **then**
    decrease max_p *as per ARED mechanism*
    $max_p = max_p \times \beta$
  end
  else if $newavg < oldavg$ **then**
    decrease max_p *as per Re-ARED-M1 mechanism*
    $\beta = 1 - \left(0.17 \times \frac{target_{low} - newavg}{target_{low} - min_{th}}\right)$
    $max_p = max_p \times \beta$
  end
end
else if $newavg > target_{up}$ **and** $max_p \leq 0.5$ **then**
  if $newavg > oldavg$ **then**
    increase max_p *as per Re-ARED-M1 mechanism*
    $\alpha = 0.25 \times max_p \times \frac{newavg - target_{up}}{target_{up}}$
    $max_p = max_p + \alpha$
  end
  else if $newavg < oldavg$ **then**
    increase max_p *as per ARED mechanism*
    $\alpha = min[0.01, 0.25 \times max_p]$
    $max_p = max_p + \alpha$
  end
end

The design of CARED algorithm gives robust performance in a wide range of environments because it combines the advantages of both ARED and Re-ARED-M1 in appropriate scenarios. The robustness of CARED algorithm comes by efficiently controlling packet drop rate during *critical time period* to maximize the overall throughput of the network. Moreover, CARED algorithm is designed to operate either like ARED or like Re-ARED-M1 depending of *level of traffic load* and hence in the worst possible scenario, performance of CARED will resort to that of ARED or Re-ARED-M1.

Unlike other RED based algorithms, CARED algorithm does not introduce new parameters to achieve performance gain. Based on *newavg* and *oldavg* values, CARED algorithm infers the *level of traffic load* and varies max_p accordingly. It must be noted that the decision to vary max_p either conservatively or aggressively in CARED, depends only on *newavg* and *oldavg* (calculated as shown in (1)) and not on max_p. Since there are only algorithmic changes and no new parameter settings in CARED, it can be easily deployed in routers without any additional complexity.

5. Results

In this section we examine two major aspects of CARED algorithm as compared to RED, ARED and Re-ARED: (i) average

queue length dynamics during *critical time period* and (ii) packet drop rate and throughput.

5.1. Average queue length dynamics during critical time period

To analyze the average queue length dynamics of CARED algorithm during *critical time period*, we simulate a dumbbell topology analogous to the one designed in Floyd et al. (2001). This topology illustrates the effect of sharp change in the congestion level on the *avg* of RED, ARED, Re-ARED and CARED algorithms. The simulation parameters are shown in Table 5.

Fig. 4(a) through (d) demonstrates average queue length dynamics of RED, ARED, Re-ARED and CARED respectively. When a sharp increase in the congestion level occurs at time 25 s, average queue length (*avg*) increases quickly. To avoid performance degradation of the network during this *critical time period*, max_p must be rapidly increased to bring *avg* within desired thresholds. Since RED does not vary max_p, as shown in Fig. 4(a), average queue length does not come back within the desired thresholds. ARED increases max_p and hence, as shown in Fig. 4(b), brings *avg* within the *target range*. Note that it takes 10 s for ARED to bring *avg* within the *target range* because it increases max_p conservatively. Unlike ARED, Re-ARED increases max_p aggressively and thus, as shown in Fig. 4(c), takes only 8 s to control *avg*. CARED increases max_p either conservatively or aggressively based on *the level of traffic load* and hence, as shown in Fig. 4(d), performs slightly better than Re-ARED.

Fig. 5(a) through (d) demonstrates average queue length dynamics of RED, ARED, Re-ARED and CARED respectively. When a sharp decrease in congestion level occurs at time 25 s, *avg* decreases quickly. To avoid under utilization of network resources during this *critical time period*, max_p must be rapidly decreased to bring *avg* within desired thresholds. Lack of variation in max_p leads to severe degradation in the performance of RED. As shown in Fig. 5(a), *avg* of RED does not come within the desired range throughout the simulation. ARED multiplicatively decreases max_p to bring *avg* within the desired *target range*. Since ARED uses a fixed value of β to decrease max_p, it takes 18 s to control *avg* (see Fig. 5(b)). Re-ARED adapts max_p based on the ratio of the change in the average queue size and hence takes only 15 s to control *avg* (see Fig. 5(c)). CARED takes only 9–10 s to bring *avg* back to *target range* because it efficiently adapts max_p based on *the level of traffic load*.

Table 5
Simulation parameters.

Parameters	Value
Bottleneck bandwidth	1.5 Mbps
Bottleneck capacity in packets	35
Mean packet size	1500 bytes
min_{th} for RED	5
max_{th} for RED	15
w_q for RED	0.0027

5.2. Packet drop rate and throughput of CARED

We demonstrate the performance gain achieved by CARED algorithm in terms of packet drop rate and throughput by simulating dumbbell topology as shown in Fig. 1 and retaining the simulation parameters as described in Section 3. Tables 6 and 7 demonstrate the packet drop rate and throughput obtained for CARED algorithm as compared to that of RED, ARED and Re-ARED

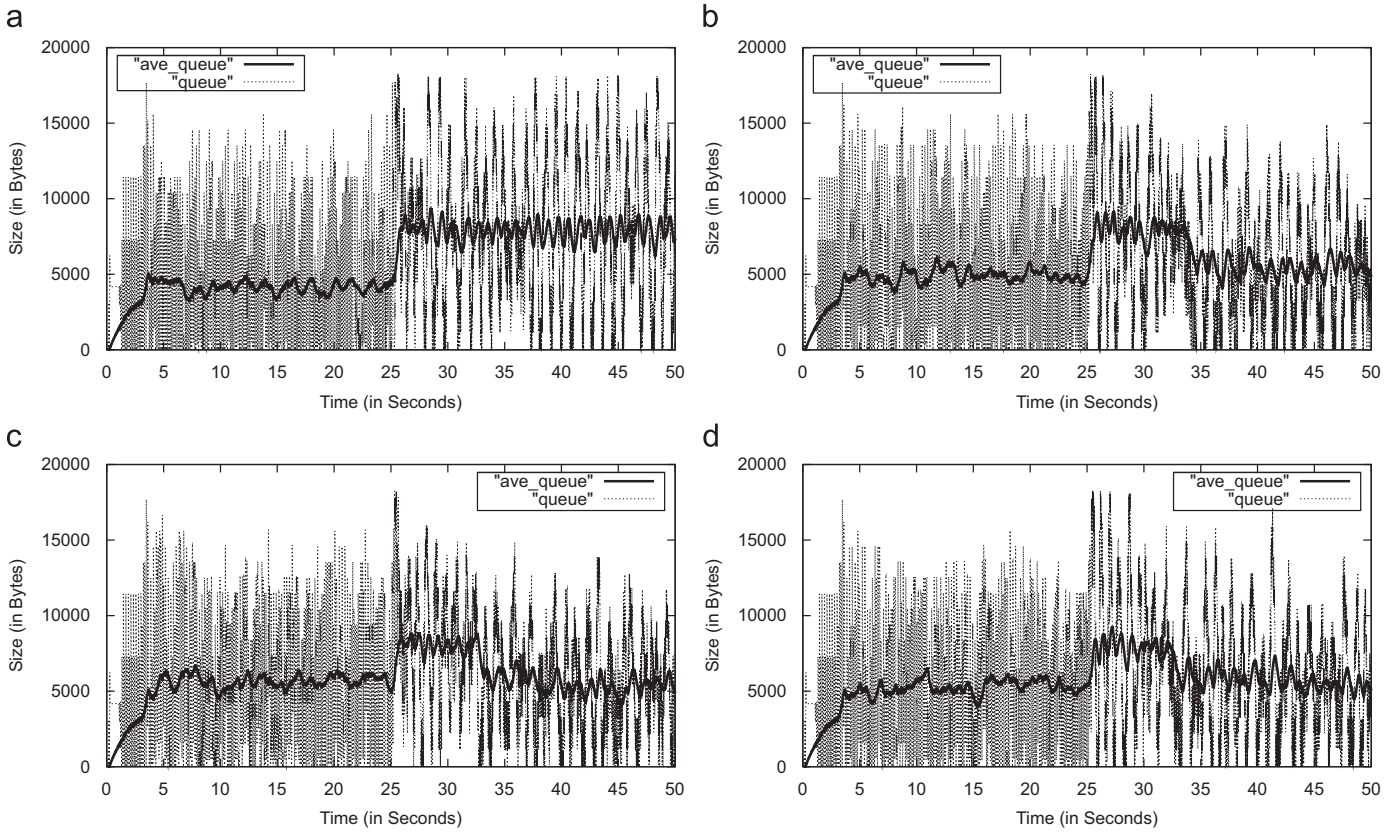

Fig. 4. *avg* with sharp increase in congestion level. (a) RED with increase in congestion level, (b) ARED with increase in congestion level, (c) Re-ARED with increase in congestion level, (d) CARED with increase in congestion level.

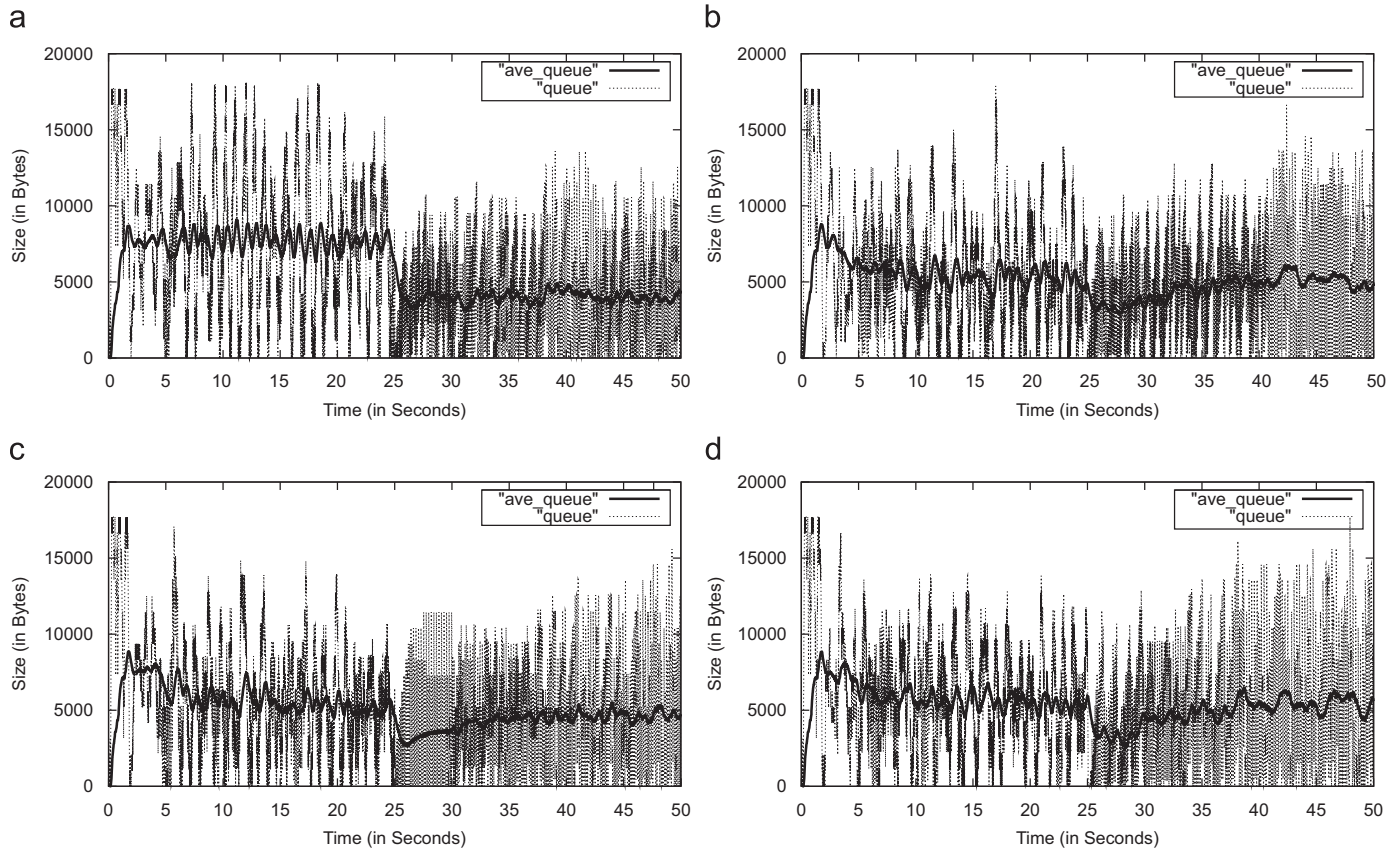

Fig. 5. *avg* with sharp decrease in congestion level. (a) RED with decrease in congestion level, (b) ARED with decrease in congestion level, (c) Re-ARED with decrease in congestion level, (d) CARED with decrease in congestion level.

Table 6
Packet drop rate of RED, ARED, Re-ARED and CARED.

FTP connections	RED (%)	ARED (%)	Re-ARED (%)	CARED (%)	Imp1 (%)	Imp2 (%)
5	0.22	0.22	0.23	0.22	0.00	0.01
10	0.25	0.23	0.22	0.23	0.00	−0.01
20	0.25	0.27	0.27	0.26	0.01	0.01
30	2.08	3.50	3.31	2.99	0.51	0.32
40	5.67	8.21	8.25	8.13	0.08	0.12
50	8.60	10.41	10.54	10.35	0.06	0.19
60	11.97	12.49	12.77	12.34	0.15	0.43
70	13.69	13.94	14.02	14.01	−0.07	0.01
80	15.47	15.67	15.29	15.46	0.21	−0.17
90	16.48	17.00	16.75	16.95	0.05	−0.20
100	17.99	17.84	17.80	17.66	0.18	0.14

Table 7
Throughput (kbps) of RED, ARED, Re-ARED and CARED.

FTP connections	RED	ARED	Re-ARED	CARED	Imp1 (%)	Imp2 (%)
5	8510	9145	9199	9181	0.39	−0.20
10	9105	9177	9189	9189	0.13	0.00
20	9181	9171	9182	9200	0.32	0.20
30	9188	9175	9186	9195	0.22	0.10
40	9133	9186	9185	9195	0.10	0.11
50	9087	9178	9178	9190	0.13	0.13
60	9083	9166	9171	9178	0.13	0.08
70	9074	9162	9160	9164	0.02	0.04
80	9069	9144	9153	9159	0.16	0.07
90	9077	9145	9139	9145	0.00	0.07
100	9056	9108	9097	9110	0.02	0.14

respectively. Imp1 and Imp2 columns represent the improvement in CARED algorithm as compared to ARED and Re-ARED respectively.

Results shown in Tables 6 and 7 confirm that the performance of RED is highly sensitive to appropriate setting of max_p. The major reason for performance degradation of ARED and Re-ARED in light and heavy traffic load scenarios respectively is, conservative adaptation of max_p by ARED and aggressive adaptation of max_p by Re-ARED during the *critical time period*.

We observe that since CARED algorithm takes into consideration *the level of traffic load* to switch from ARED to Re-ARED and vice versa, it efficiently controls the packet drop rate and maximizes the throughput of the network. It is observed that CARED reduces the packet drop rate upto 0.51% and 0.43% as compared to ARED and Re-ARED respectively. The improvement in throughput achieved by CARED algorithm is upto 0.39% and 0.2% as compared to ARED and Re-ARED respectively.

The effectiveness of CARED algorithm comes by cautiously adapting max_p during *critical time period* (see Section 5.1). As a result, the performance of CARED is robust in a wide variety of environments. Though CARED algorithm does not achieve least packet drop rate and/or maximum throughput in some scenarios, its performance in such scenarios is always between ARED and Re-ARED. This is because CARED is designed to operate either like ARED or like Re-ARED-M1 depending on the *level of traffic load*.

6. Conclusions

In this paper we show that ARED is conservative in adapting max_p and hence leads to degradation of throughput when level of

congestion varies abruptly. Though Re-ARED addresses the drawback of ARED and varies max_p aggressively, it does not provide robust performance when the traffic load is heavy. Hence we have proposed two modifications to Re-ARED algorithm to make it more robust especially when traffic load is heavy. Moreover, we have designed CARED algorithm that combines ARED and our modified Re-ARED based on the *level of traffic load*. Simulation results show that CARED algorithm efficiently controls the packet drop rate and improves the overall throughput of the network. CARED algorithm combines the advantages of both ARED as well as Re-ARED and hence provides robust performance in a wide range of environments. Since CARED algorithm does not introduce new parameters to achieve performance gain, it can be deployed without any additional complexity and its performance can be further evaluated on a real time testbed.

Recent work has focused on leveraging the benefits of AQMs to improve the performance of wireless networks. Apart from congestion, interference is also an important factor that leads to queue build up in wireless networks. The performance of CARED can be further analyzed in wireless networks by studying the benefits of considering *level of traffic load* as a parameter to vary maximum drop probability.

References

Abbasov B, Korukoglu S. Effective RED: an algorithm to improve RED's performance by reducing packet loss rate. Journal of Network and Computer Applications 2009;32:703–9.

Aweya J, Ouellette M, Montuno DY. A control theoretic approach to active queue management. Computer Networks 2001;36:203–35.

Chen, J., Hu, C., Ji, Z. Self-tuning random early detection algorithm to improve performance of network transmission. Mathematical Problems in Engineering, doi:10.1155/2011/872347, in press. (Article ID 872347).

Feng G, Agarwal A, Jayaraman A, Siew CK. Modified RED gateways under bursty traffic. IEEE Communications Letters 2004;8:323–5.

Feng W-C, Kandlur D, Saha D, Shin K. A self-configuring RED gateway. in: Proceedings of IEEE INFOCOM'99. Eighteenth annual joint conference of the IEEE Computer and Communications Societies, vol. 3; 1999. p. 1320–8.

Feng W-C, Kandlur DD, Saha D, Shin KG. Blue: an alternative approach to active queue management. in: Proceedings of the 11th international workshop on network and operating system support for digital audio and video (NOSSDAV '01). NY, USA: Port Jefferson; 2001. p. 41–50.

Floyd S, Gummadi R, Shenker S. Adaptive RED: an algorithm for increasing the robustness of RED's active queue management. Technical report; 2001.

Floyd S, Jacobson V. Random early detection gateways for congestion avoidance. IEEE/ACM Transactions on Networking 1993;1:397–413.

Gettys J. Bufferbloat: dark buffers in the internet. IEEE Internet Computing 2011;15:96.

Hariri B, Sadati N. Nn-RED: an AQM mechanism based on neural networks. Electronics Letters 2007;43:1053–5.

Javam H, Analoui M. Sared: stabilized ARED. in: International conference on communication technology, 2006, ICCT'06; 2006. p. 1–4.

Kim T-H, Lee K-H. Refined adaptive RED in TCP/IP networks. in: International joint conference, SICE-ICASE, 2006; 2006. p. 3722–5.

Kunniyur S, Srikant R. Analysis and design of an adaptive virtual queue (AVQ) algorithm for active queue management. SIGCOMM Computer Communication Review 2001;31:123–34.

Lakshman TO, Lakshman TV, Wong L. Sred: stabilized RED. in: Proceedings of INFOCOM; 1999. p. 1346–55.

Liu S, Basar T, Srikant R. Exponential-RED: a stabilizing AQM scheme for low- and high-speed TCP protocols. IEEE/ACM Transactions on Networking 2005;13:1068–81.

Mahajan R, Floyd S. Controlling high bandwidth flows at the congested router. in: Proceedings of IEEE ICNP'01. IEEE; 2001.

Marquez R, González I, Carrero N, Sulbarán Y. Revisiting adaptive RED: beyond AIMD algorithms. in: Proceedings of the 1st EuroFGI international conference on network control and optimization NET-COOP'07. Berlin, Heidelberg: Springer-Verlag; 2007. p. 74–83.

Misra V, Gong W-B, Towsley D. Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED. SIGCOMM Computer Communication Review 2000;30:151–60.

UCN/LBL/VINT. Network simulator-2 (ns-2). Available from 〈http://www.isi.edu/nsnam/ns/〉; 2011.

Verma R, Iyer A, Karandikar A. Active queue management using adaptive RED. IEEE/KICS Journal of Communications and Networks 2002:5.

Wang C, Liu J, Li B, Sohraby K, Hou YT. Lred: a robust and responsive AQM algorithm using packet loss ratio measurement. IEEE Transactions on Parallel and Distributed Systems 2007;18:29–43.

Weighted Random Early Detection (WRED). Available from 〈http://www.cisco.com/en/US/docs/ios/11-2/feature/guide/wred-gs.pdf/〉; 2009.

Zheng B, Atiquzzaman M. Dsred: an active queue management scheme for next generation networks. in: Proceedings of the 25th annual IEEE conference on local computer networks, LCN'00. Washington, DC, USA: IEEE Computer Society; 2000.

Zhou K, Yeung KL, Li VOK. Nonlinear RED: a simple yet efficient active queue management scheme. Computer Networks 2006;50:3784–94.