37th Annual Simulation Symposium, Hyatt Regency Crystal City, Arlington, VA, April 18 - 22, 2004

Markov Model Based Congestion Control for TCP

Shan Suthaharan
University of North Carolina at Greensboro, Greensboro, NC 27402, USA
ssuthaharan@uncg.edu

Abstract

The Random Early Detection (RED) scheme for congestion control in TCP is well known over a decade. Due to a number of control parameters in RED, it cannot make acceptable packet-dropping decision, especially, under heavy network load and high delay to provide high throughput and low packet loss rate. We propose a solution to this problem using Markov chain based decision rule. We modeled the oscillation of the average queue size as a homogeneous Markov chain with three states and simulated the system using the network simulator software NS-2. The simulations show that the proposed scheme successfully estimates the maximum packet dropping probability for Random Early Detection. It detects the congestion very early and adjusts the packet-dropping probability so that RED can make wise packet-dropping decisions. Simulation results show that the proposed scheme provides improved connection throughput and reduced packet loss rate.

1. Introduction

The Random Early Detection (RED), an active queue management technique, has been suggested as a solution to solve network congestion problem in TCP/IP networks [1]. However, it does not provide an acceptable solution because of its dependency on a number of control parameters such as *wq* (weight used in exponential weighted averaging), *th_min*, *th_max* (minimum and maximum thresholds used for queue management) and *max_p* (maximum packet-dropping probability) and unpredictable system parameters such as round-trip time of network connections, load of the network (flows or connections). RED is recommended by the Internet Engineering Task Force (IETF) for use in the routers of Next Generation [2], therefore, our goal in this paper is to solve this problem with minimal changes to the overall RED algorithm.

RED manages the queue dynamically by randomly dropping packets with increasing probability as the average queue size increases. It makes the packet-dropping decision at time *t* based on the dynamics and the state of the average queue size at time *t*. It increases the packet drop rate linearly from zero (as the average queue size at *th_min*) to a drop rate of *max_p* (when the average queue size reaches *th_max*). Thus the average queue size plays a major role in the RED's packet-dropping decision. The idea behind RED active queue management technique is to detect congestion early and send early congestion notification to the end-hosts so that they can adjust their transmission rate to reduce packet loss and increase overall throughput. However, the main weakness of RED is that the instability caused by the highly oscillating average queue size [3]. The oscillation is caused by the difficult nature of selecting accurate RED's control parameters and unpredictable system parameters, such as amount of network traffic and the round-trip time (RTT) of network connections [4]. This oscillation problem leads to decreased throughput and increased packet-dropping rates.

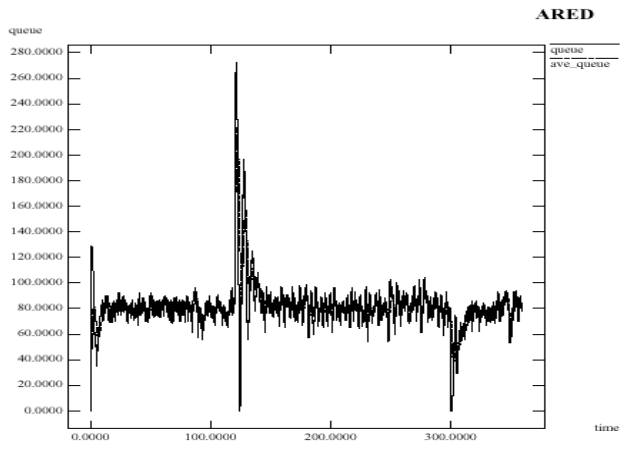
An Adaptive Random Early Detection (ARED) algorithm has been recently suggested to address this problem [5, 6]. The ARED dynamically updates the maximum packet-dropping rate *max_p* in order to keep the average queue size closer to a target queue size that may be chosen by the system administrator to compromise between low delay and high utilization. In general, the target queue size is selected at half way between the thresholds *th_min* and *th_max*. The average queue size and the target queue size are compared once every small interval Δt. If the average queue size is greater than the target queue size and the maximum packet-dropping probability *max_p* is less than or equal to 0.5, the *max_p* is increased by a constant α. Similarly if the average queue size is less than the target queue size and the *max_p* is greater than or equal to 0.01 the *max_p* is decreased by β. It uses the AIMD (additive-increase-multiplicative-decrease) policy to increase and decrease the *max_p* as appropriate. The ARED maintains average queue size and provides reduced parameter sensitivity in the steady-state and low varying queue size conditions. The AIMD process does not consider the stochastic behavior of the queue, thus the adaptive nature of the ARED fails when a large number of flows with large varying queue size are present in a network.

As we have seen, the main idea of ARED is to adapt *max_p* to keep the average queue size with the target range between *th_min* and *th_max* at time *t*. In this paper we take a different approach to update the *max_p* at time *t* and keep the average queue size closer to the target range. We use the stochastic nature of the average queue size and its oscillation in the state space {0, 1, 2} to achieve this goal. Using three states Markov model we obtain transition probability matrix dynamically. The matrix at time t will be used to predict the average queue state at time t+1. In contrast to ARED that uses only the average queue size information at time t, the proposed stochastically adaptive random early detection (SARED) technique uses both the current average queue size and the predicted queue size information together with their corresponding transition probabilities to alter the *max_p*. Our finding is that the use of Markov model in making modification to *max_p* dynamically yields better performance than ARED and Feng schemes in terms of reducing packet loss rate and increasing throughput gain.

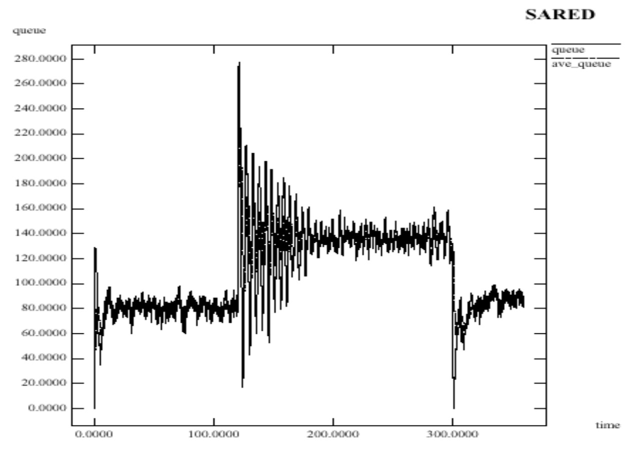
The rest of the paper is organized as follows: Section 2 presents a simple example to show the motivation behind the SARED. In section 3 we present the proposed SARED technique with two algorithms (i) for calculating transition probabilities for average queue size and (ii) for adapting *max_p* to adjust the average queue size. Section 4 presents results and findings from five simulations to show the effectiveness of SARED in terms of increasing throughput and reducing packet loss. In section 5 conclusions are presented.

2. The Motivation SARED

The purpose of the original adaptive RED [5] is to adapt the *max_p* to keep the average queue size between *min_th* and *max_th*. However, it has been recently modified to adapt the *max_p* to keep the average queue size with a target range half way between *min_th* and *max_th* [6]. The constraints of target range and the thresholds negatively impact on the packet throughput and packet loss. For example when there is a sharp change in the level of congestion, the ARED tries to bring the queue size to a target range instead of helping the RED to improve the throughput and reduce packet loss. Although we will provide simulation results later in detail, we show here two examples of an experiment in Figures 1(a) and 1(b) to help understand the motivation behind the development of stochastically adaptive RED. These figures show that a sharp increase in the level of traffic congestion at 120th seconds in the entire simulation time of 360 seconds, how well the ARED successfully bring the average queue size to the target level with throughput of 89.94%, and how well the SARED increases the average queue size to handle the significant increase in the level of traffic congestion with throughput of 91.15%. The SARED increases the throughput and decreases the packet loss by adjusting the average queue size above the target range (the same as the one used in ARED) used in the SARED. It occurs because the SARED monitors the changes in the average queue size and predicts the future changes in the queue size, and thus it adapts the *max_p* to increase connection throughput and reduce packet loss.



(a)



(b)

Figure 1: (a) ARED with an Increase in Congestion. (b) SARED with an Increase in Congestion.

3. Stochastically Adaptive RED

The RED makes the packet-dropping decision at time t based on the dynamics and the state of the average queue size at time t. The average queue size can be in one of the possible states 0, 1 and 2 at time t, where 0 indicates the average queue size is in the interval [0, *th_min*], the state 1 indicates the average queue size is in the interval (*th_min*, *th_max*), and the state 2 indicates the average queue size is in the interval [*th_max*, buffer_size]. The RED increases the packet drop rate linearly from zero (as the average queue size at *th_min*) to a drop rate of *max_p* (when the average queue size reaches *th_max*). The ARED and the proposed SARED attempt to modify the *max_p* appropriately. In the SARED, which adapts the stochastic behavior of the average queue size, the queue management system (QMS) observes the stochastic behavior of the average queue size at a discrete set of times $t_0, t_1, t_2 \ldots t, \ldots\ldots$, where $t_i - t_{i-1}$ is very small. Let the successive observations of *average queue size* be denoted by $X_0, X_1, \ldots. X_t, \ldots.$ and therefore it can be assumed that X_t is a random variable. The value of X_t represents the state {0, 1 or 2} at time t of the system. Therefore, the system has (i) a finite number of states, (ii) X_t is a random variable and (iii) X_t depends on X_{t-1} and thus the sequence $\{X_t\}$ is a Markov chain [7]. Our approach is based on the RED algorithm, therefore the average queue size at time t_i depends on the average queue size at time t_{i-1}, and we approximate the behavior of the average queue size X_t to a homogeneous Markov chain within a small interval. Its probability mass function and 1-step transition probability function are defined respectively as follows:

$p_j(t) = P[X_t = j]$, $p_{jk}(1) = P[X_{t+1} = k \mid X_t = j]$, for any integer $t \geq 0$, where $j, k \in \{0, 1, 2\}$.

We simply denote these probabilities as p_j and p_{jk} respectively. Using the notations, the transition probabilities of the Markov chain $\{X_t\}$ with state space {0, 1, 2} is exhibited in the following matrix form:

$$P(1) = \begin{bmatrix} p_{00} & p_{01} & p_{02} \\ p_{10} & p_{11} & p_{12} \\ p_{20} & p_{21} & p_{22} \end{bmatrix}$$

One of the main goals of ARED and SARED is to keep the oscillation in the queue size within a range, thus we assume if the system is in a particular state it has high probability to be within the same state for a long period of time. Therefore, the following initial 1-step transition probability has been chosen:

$$P(1) = \begin{bmatrix} 0.9 & 0.05 & 0.05 \\ 0.05 & 0.9 & 0.05 \\ 0.05 & 0.05 & 0.9 \end{bmatrix}$$

Similarly our goal is to keep the average queue size in the middle range between the thresholds *th_min* and *th_max*, therefore, $p_0 = 0.05$, $p_1 = 0.9$, $p_2 = 0.05$ are used. The QMS calculates the 1-step transition probability matrix dynamically at time t_i before making its packet-dropping decisions using the stochastic behavior of the average queue size in the predetermined interval $[t_{i-1}, t_i]$. The small interval makes the homogeneous assumption of the average queue size valid:

```
if (old_ave < th_min && new_ave < th_min)
    n00 = n00 + 1.0;
if (old_ave < th_min && (new_ave >= th_min
&& new_ave < th_max))
    n01 = n01 + 1.0;
if (old_ave < th_min && new_ave >= th_max)
    n02 = n02 + 1.0;
if ((old_ave >= th_min && old_ave < th_max)
&& new_ave < th_min)
    n10 = n10 + 1.0;
if ((old_ave >= th_min && old_ave < th_max)
&& (new_ave >= th_min && new_ave < th_max))
    n11 = n11 + 1.0;
if ((old_ave >= th_min && old_ave < th_max)
&& new_ave >= th_max)
    n12 = n12 + 1.0;
if (old_ave >= th_max && new_ave < th_min)
    n20 = n20 + 1.0;
if (old_ave >= th_max && (new_ave >= th_min
&& new_ave < th_max))
    n21 = n21 + 1.0;
if (old_ave >= th_max && new_ave >= th_max)
    n22 = n22 + 1.0;

n0 = n00+n01+n02
n1 = n10+n11+n12
n2 = n20+n21+n22

p00 = n00/n0; p01 = n01/n0; p02 = n02/n0;
p10 = n10/n1; p11 = n11/n1; p21 = n21/n1;
p20 = n20/n2; p21 = n21/n2; p22 = n22/n2;
```

In this algorithm n_{ij} represents the number of times the average queue size is changed from state i to state j from the time the queue management start monitoring the queue to the current time, where i, j = 0, 1 or 2. Unlike ARED, which makes its packet-dropping decision at t_i based only on the current average queue size at time t_i, the proposed SARED uses both current average queue size at t_i and the predicted queue status information (transition probabilities generated using the stochastic behavior of the average queue size in the interval $[t_{i-1}, t_i]$) at t_{i+1} to make packet dropping decision at t_i. The SARED algorithm is given below:

```
if (now > lastset + interval) {
  if (ave <= th_min+part) {
    if (max_p <= 0.01) {
     alpha=p01*(p0+p2); beta=p00*p1;
     max_p=max_p*beta+alpha; lastset=now;}
    else if (max_p >= 0.5) {
     beta=(p22-p21-p10)*p1;
     max_p=max_p*beta; lastset=now;}
    else {
     beta=(p11-p10)*p1;
     max_p=max_p*beta; lastset=now;}
  }
  else
  if (ave>th_min+part && ave<th_max-part){
    if (max_p <= 0.01){
     alpha=p01*(p0+p2);
     max_p=max_p+alpha; lastset=now;}
    else if (max_p >= 0.5){
     beta=(p22-p21)*p1;
     max_p=max_p*beta; lastset=now;}
    else {
      alpha=$p_{12}*(p_0+p_2)$; beta=$p_{11}*p_1$;
      max_p=max_p*beta+alpha; lastset=now;}
  }
  else if (ave >= th_max-part){
    if (max_p <= 0.01){
      alpha=$(p_{01}+p_{12})*(p_0+p_2)$;
      max_p=max_p+alpha; lastset=now;}
    else if (max_p >= 0.5)
    {
      alpha=$p_{21}*(p_0+p_2)$; beta=$p_{22}*p_1$;
      max_p=max_p*beta+alpha; lastset=now;}
    else {
      alpha=$p_{12}*(p_0+p_2)$;
      max_p=max_p+alpha; lastset=now;}
  }
}
```

In this algorithm the variable *part* is set to *0.4**(*th_max-th_min*) to keep the average queue size half way between *th_min* and *th_max* in a steady state situation and to keep the queue size closer to the range [*th_min, th_max*] when there is a sudden increase in the level of congestion. The *interval* set to 0.5, which is kept small to satisfy the homogeneous property of the Markov model. The motivation behind this SARED algorithm is to adapt the *max_p* to remain closer to the range [0.01, 0.5] based on the probabilities that a state can be visited, and that the average queue size can move from one state to another. For example if the average queue size is in the state 0 (meaning less than *th_min*+*part*) and the current *max_p* is greater than 0.5 the *max_p* should be reduced and this will be done by multiplying the current max_p by $(p_{22}-p_{21}-p_{10})*p_1$ to let the source to increase the transmission rate. With similar reasons other multiplicative (beta) and additive (alpha) factors are determined as shown in the above algorithm. The statement `lastset=now` and the condition `now > lastset + interval` allow the QMS to calculate transition probabilities before making packet dropping decision.

4. Simulations

We present results and findings from *five* different simulations to demonstrate: (i) the effectiveness of our proposed SARED in improving packet throughput and reducing packet loss when there is a sudden change in the level of congestion with small amount of tcp traffic (ii) the effectiveness of our proposed SARED in improving packet throughput and reducing packet loss when there is a sudden change in the level of congestion with large amount of tcp traffic (iii) some oscillation problems when small amount of traffic present (iv) the effectiveness of our proposed SARED when the system is in steady-state (meaning there is no sudden change in the traffic level) and (v) the effectiveness of the proposed SARED compared with Feng and ARED schemes. The simulations are conducted using ns-2 [8]. The main purpose of this paper is to propose a better technique than ARED and allow this technique to be deployed with RED implementations in the next generation Internet routers. Also our intention is not to show that SARED is optimal and therefore we didn't compare the results with the BLUE [9], which in general gives better throughput and reduced packet loss in the experimental scenarios considered in this paper.

4.1 First Simulation

In this simulation we use the network configuration shown in Figure 2. The RED's control parameters employed at the bottleneck router R_1 for this simulation are as follows: *th_min* = 40, *th_max* = 120, *buffer_size* = 280, *wq* = 0.002 and *max_p* = 0.125. For RED and ARED, the recommendations are for *th_max* to be set to three times *th_min* [5]. The duration of the simulation is 360 sec and the mean packet size is 1500.

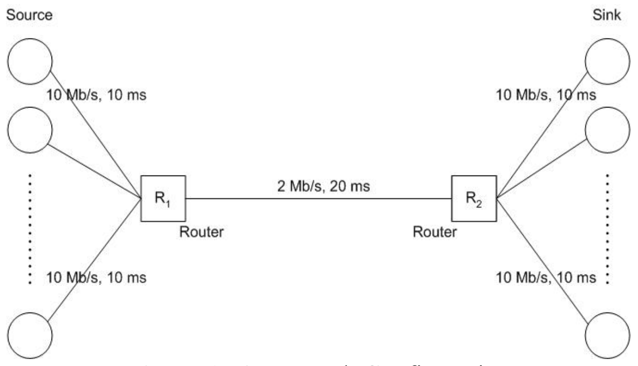
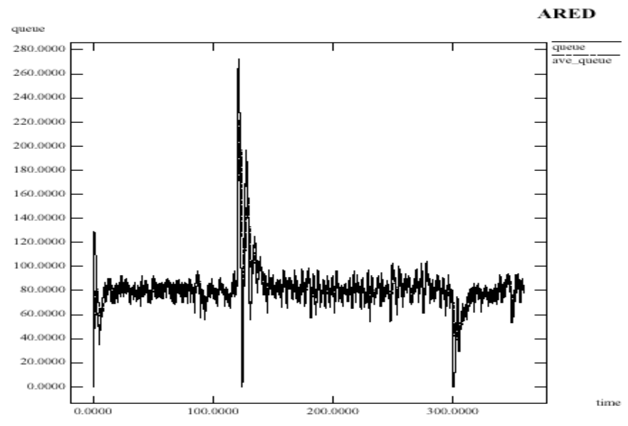
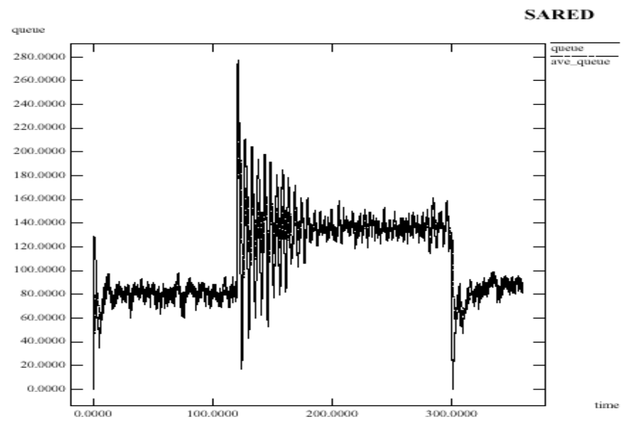


Figure 2: A Network Configuration

In the simulation we have 10 tcp flows running until the entire duration of 360 seconds, but 90 more tcp flows started at 120 seconds to make sudden change in the congestion level and they last for next 180 seconds. The queue length and average queue length results using ARED and SARED are shown in Figures 3(a) and 3(b) respectively.



(a)



(b)

Figure 3: (a) ARED with an Increase in Congestion using Medium No. of TCP-flows. (b) SARED with an Increase in Congestion using Medium No. of TCP-flows.

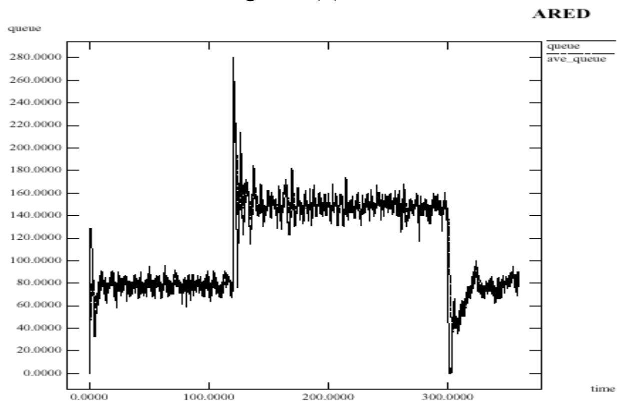
We observed 9680 packets lost and 86576 packets sent with ARED simulation shown in Figure 3(a) whereas we observed 8410 packets lost and 86586 packets sent with SARED simulation shown in Figure 3(b). Our finding from this experiment is that when a network is congested with a small amount of traffic the ARED maintains the average queue size within the target range as expected. However, SARED adaptively pushes the average queue size to above the target range to increase the throughput and reduce the packet loss and this is a preferred scenario.

4.2 Second Simulation

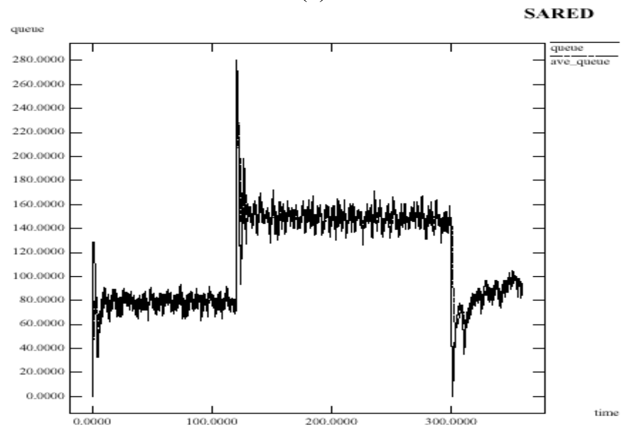
In this simulation we have 10 tcp flows running until 360 seconds and 210 more tcp flows started suddenly and last for next 180 seconds. The results are shown in Figures 4(a) and 4(b). We observed 13316 packets lost and 86679 packets sent in the ARED simulation shown in Figure 4(a), whereas we observed 13264 packet lost and 86692 packets sent with SARED simulation shown in Figure 4(b). Our finding from this experiment is that when a network is congested with a large amount of traffic, the SARED and ARED behaves the same way in terms of controlling the average queue size as shown in Figures 4(a) and 4(b), however, the SARED gives slightly better results with increased throughput and reduced packet loss. It can be clearly seen, by comparing Figure 3(a) and Figure 4(a), that the ARED lost its purpose of maintaining the average queue length within the target range when a large amount of increase in the congestion level.

4.3 Third Simulation

In this simulation we used the same network configurations as per previous simulations, but this time we used 50 more tcp flows instead of 90 or 210. The results are shown in Figures 5(a) and 5(b). We observed 6519 packets lost and 86530 packets sent with ARED simulation shown in Figure 5(a) whereas we observed 5192 packets lost and 86572 packets sent with SARED simulation shown in Figure 5(b).



(a)



(b)

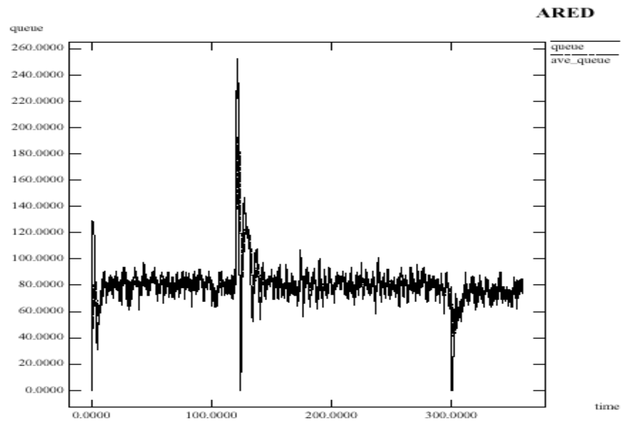
Figure 4: (a) ARED with an Increase in Congestion using Large No. of TCP-flows. (b) SARED with an Increase in Congestion using Large No. of TCP-flows.

Our finding from this experiment is that when very small amount of traffic involve in changing the congestion suddenly, there is not enough data for the SARED to calculate the transition probabilities to predict the average queue size and adapt the *max_p* appropriately, thus it oscillates a lot to correct the *max_p*. However, it still (i) increases the throughput (ii) reduce the packet loss and (iii) pushes the average queue length to deal with the sudden change in the traffic to improve the throughput and reduce the packet loss.

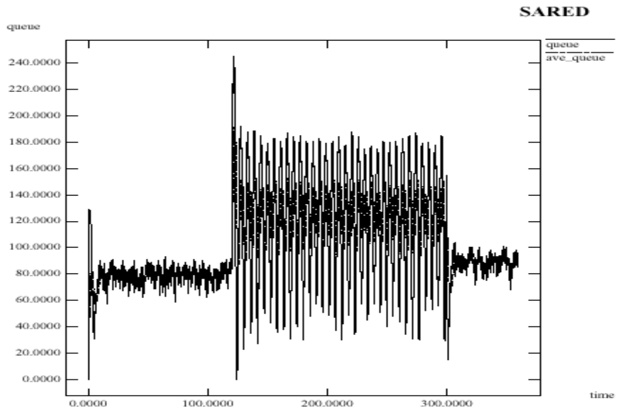
4.4 Fourth Simulation

In this simulation also we used the same network configurations as per previous simulations, but this time we used steady-state congestion with 60 and 140 tcp flows. The results are shown in Figures 6(a) and 6(b), and

Figures 7(a) and 7(b) respectively. We obtained the following results for 60 tcp traffic: we observed 12341packets lost and 86577 packets sent with ARED simulation shown in Figure 6(a) whereas we observed 9773 packets lost and 86577 packets sent for SARED as shown in Figure 6(b). Similarly, we obtained the following results for 140 tcp traffic: we observed 21594 packets lost and 86657 packets sent with ARED simulation shown in Figure 7(a) whereas we observed 20119 packets lost and 86657 packets sent for SARED as shown in Figure 7(b). Our finding from this experiments is that in steady-state scenario once again the ARED concentrates on controlling average queue size within the target range whereas the SARED pushes the average queue size up, but closer to the target range so that increased throughput and reduced packet loss can be obtained.



(a)

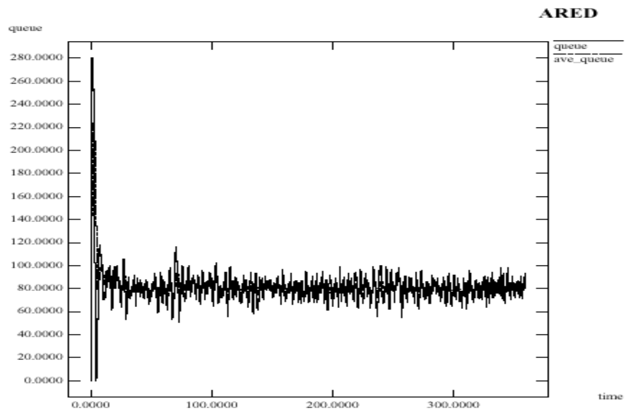


(b)

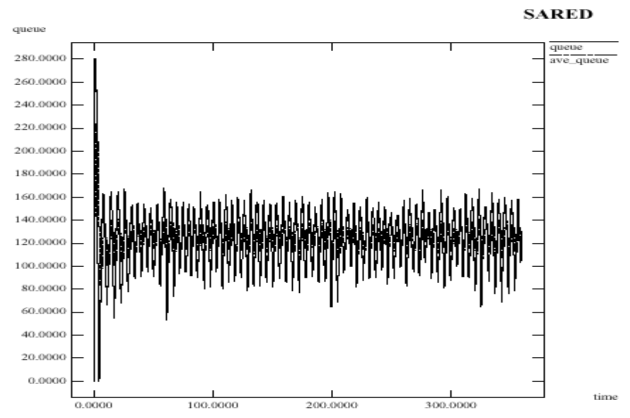
Figure5: (a) ARED with an Increase in Congestion using Large No. of TCP-flows. (b) SARED with an Increase in Congestion using Large No. of TCP-flows.

4.5 Fifth Simulation

The fifth simulation is two folded: (i) is to compare the performance of Feng, ARED and SARED techniques in terms of throughput gain and packet-loss rate over different number of TCP flows (connections) and (ii) is to compare the performance of these three methods over different delay (round-trip time) in the outbound link of the bottleneck router. The network configuration we used is shown in Figure. 8 in which the bandwidth and delay of each full duplex link are illustrated.

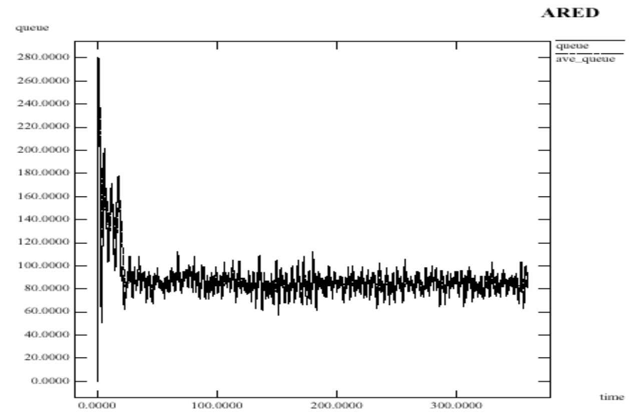


(a)

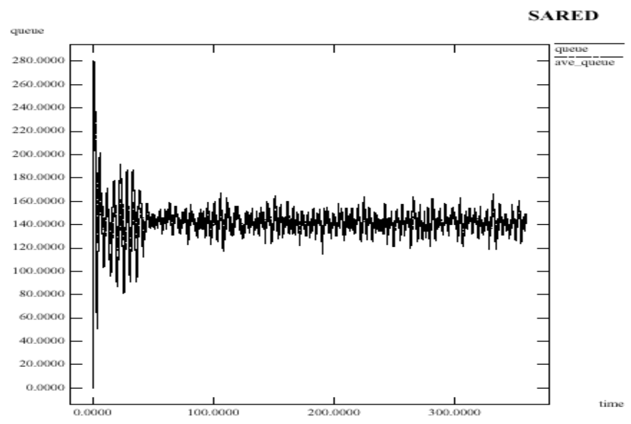


(b)

Figure 6: (a) ARED with steady-state. (b) SARED with steady-state.



(a)



(b)

Figure 7: (a) ARED with steady-state. (b) SARED with steady-state.

The RED's control parameters employed at the bottleneck router R_1 for both simulations are as follows: *th_min* = 20, *th_max* = 60, *buffer_size* = 120, *wq* = 0.002 and *max_p* = 0.0. The duration of the simulation is 60 sec. In the first simulation, we conducted 20 tests using {5, 10, 15, 20, 25, 30 … 95, 100} connections as source and sink with the bandwidth of 10 Mb/s and 10ms delay for Feng, ARED and SARED techniques. The bottleneck link has 5 Mb/s bandwidth and 20 ms delay. The throughput gain and packet loss percentages are recorded for each test for the three techniques. The results are presented in Figures 9(a) and 9(b). They show that the throughput gain of SARED is higher than that of the Feng and ARED schemes. Similarly, the packet loss rate is significantly low for SARED than ARED and Feng schemes.

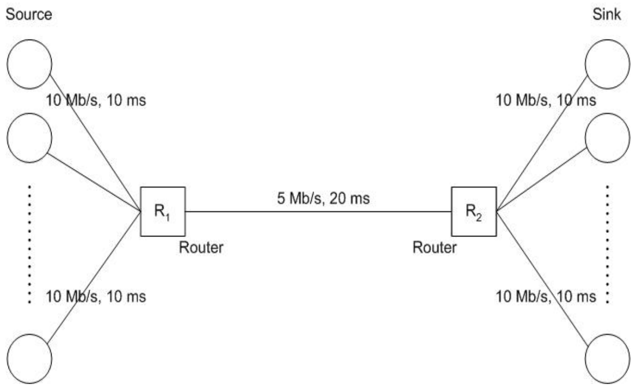
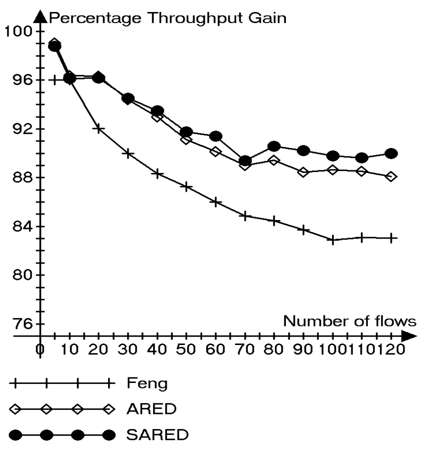
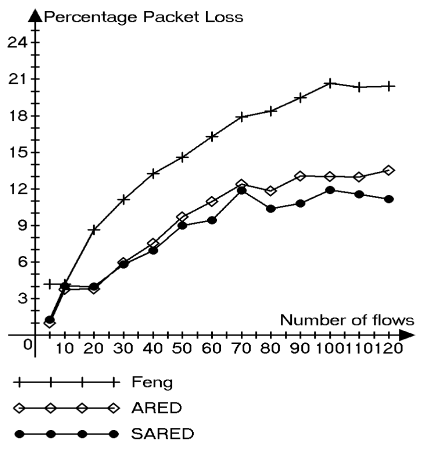


Figure. 8: Network configuration for the simulations.

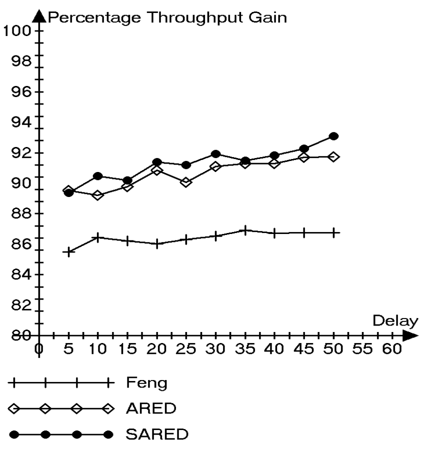
In the next simulation, we test the performance of all three methods over different outbound link delay and TCP flows. Only the results of up to 60ms delays are presented in Figure 9(c) and 9(d). However, the results are similar for other cases. They show that the throughput gain is increased and packet loss rate is decreased by the SARED compared to Feng and ARED schemes. The above simulations show that the ARED provides slightly better performance than SARED in the low network load and low delay as expected. However, SARED provides significantly better performance in the real network scenario where in general network loads and delay are high.



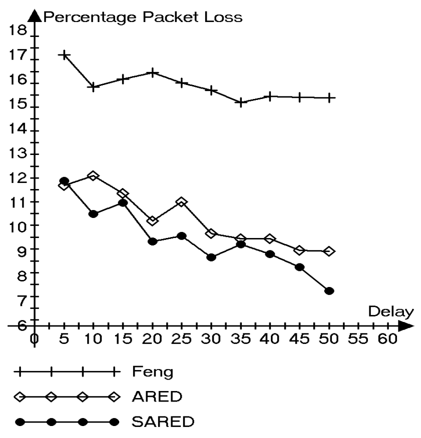
(a)



(b)



(c)



(d)

Figure 9: (a) Throughput gain comparison between Feng, ARED and SARED over different flow sizes. (b) Packet loss rate comparison between Feng, ARED and SARED over different flow sizes. (c) Throughput gain comparison between Feng, ARED and SARED with different delay. (d) Packet loss rate comparison between Feng, ARED and SARED with different delay.

5. Conclusions

This paper presents a new technique called Stochastically Adaptive RED for congestion control in TCP/IP networks. It models the behavior of the average queue size as the Markov model with three states and uses the ARED's recommendation of automatically setting the RED's parameters w_q and *th_max* to improve connection throughput with reduced packet loss. The model provides transition probabilities to predict the average queue size, detects the congestion very early and adjusts the packet-dropping probability so that RED can make wise packet-dropping decisions. The theory and experiments have demonstrated the ability of the Stochastically Adaptive RED algorithm to improve connection throughput and reduce packet loss. It is not our intention to claim that SARED is optimal or better than BLUE, but our experiments show that it works for wide range of scenarios and provides better technique than ARED so that it can be deployed with RED implementations in the next generation Internet routers.

6. Acknowledgement

I would like to thank Priya Ranjan for providing us a tcl-scipt, which uses random number generators. This helped us to develop the tcl-script for our simulations.

7. References

[1] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, August 1993, 1(4): 397-413.

[2] J. Heinanen, F. baker, W. Weiss, and J. Wroclawski, "Assured Forwarding PHB Group," RFC 2597, June 1999.

[3] W. Feng, D. Kandlur, D. Saha, and K Shin, "A self-configuring RED gateway," in *Proc*. IEEE INFOCOM, March 1999, pp. 1320-1328.

[4] P. Ranjan, E. H. Abed, and R. J. La, "Nonlinear instabilities in TCP-RED," in Proc. IEEE INFOCOM, 2002.

[5] W. Feng, K. Kandlur, D. Saha, and K. Shin, "Techniques for eliminating packet loss in congested TCP/IP Network," University of Michigan CSE-TR-349-97, November 1997.

[6] S. Floyd, R. Gummadi, and S. Shenker, "Adaptive RED: an algorithm for increasing the robustness of RED's active queue management," Available at http://www.icir.org/floyed, August 2001.

[7] E. Parzen, *Stochastic Process*, Holden-Day Inc., 1962.

[8] The Network Simulator: ns-2 [online]. Available at http://www.isi.edu/nsnam/ns/.

[9] W. C. Feng, K. G. Shin, D. D. Kandlur and D. Saha, "The Blue Active Queue Management Algorithms,"IEEE/ACM Transactions on Networking, August 2002, Vol. 10, No. 4, pp. 513-528.