A Congestion Avoidance Scheme Based On Buffer Occupancy

1Goree. Narsimhulu, 2 Dr.D. Sreenivasa Rao
1Research Scholar,department of ECE JNTU, Hyderabad,India
narsiroopa@gmail.com
2Professor ,Department of ECE
Jawaharlal Nehru Technological University, Hyderabad, India
dsraoece@jntuh.ac.in

Abstract— **Recently many prominent web sites face so called Distributed Denial of Service Attacks (DDoS). DDoS attacks are a virulent, relatively new type of attack on the availability of Internet services and resources. To avoid denigration most of the commercial sites do not expose that they were attacked that is the biggest challenges of the researchers. Network congestion caused by DDoS attack can be managed by AQM (Active queue Management).Random Early Detection (RED) is one of the most prominent congestion avoidance schemes in the Internet routers. To overcome the limitations of the basic RED algorithm, researchers proposed several variants of RED. For solving this problem, this paper proposes a new mechanism to improve RED algorithm, which is named BO-AURED (An Adaptive RED Algorithm Combined With Buffer Occupation and upper threshold). By matching router's buffer occupation with w_q, min_{th}, max_{th}, U_{th} and P_{max} parameter settings, to make BOAURED adapt to network environment variation automatically. Simulation is done in NS- 2.35 simulator environment. Simulation results show that our new BO-UARED algorithm gives better performance than RED and Adaptive RED. Comparisons are done in terms of total average throughput, total packet drops, and average packet drops. It will also increase adaptability of RED.**

Keywords- BOAURED algorithm, DDoS, RED algorithm, Adaptive RED algorithm, AQM, Packet Drops, Throughput.

I. INTRODUCTION

An aim of an internet is to provide scalable, open [1] and secured network. Confidentiality, authentication, message integrity and non-repudiation are the basic aspects of the internet security. Distributed denial of service (DDoS) attack targets the availability of services on the Internet. It is one kind of Denial of service attack. High bandwidth traffic aggregates may occur during times of flooding based DDoS attack.[2] This can make network congested and bring servers down with huge packets. DDoS flows that do not cut down their sending rates after their packets are dropped.[3]The defence mechanism of DDoS is one the aggregate based congestion control. Active Queue Management (AQM) algorithms are the key technology of congestion control.[3] The main focus on this research is to study Random Early Detection (RED) congestion control algorithms and also to provide effective solution to avoid congestion collapse of network services. We propose new algorithm using existing RED algorithm. We introduced new threshold U_{th}(Upper Threshold) and considering buffer occupancy of router and have modified RED algorithm. Simulation is done in NS 2.35 simulator. Simulation results are compared with RED and Adaptive RED with our proposed BO-AURED algorithm. Results are in terms of throughput and packet drops.

This paper is organized as follows. Chapter II gives the basic idea about DDoS attack while chapter III describes basic RED algorithm, RED drop function, and problems in RED. In chapter IV our proposed algorithm and BOAURED packet Drop function is explained. Chapter V simulation results and comparisons are shown.

II. RELATED WORK

DDoS attacks are a virulent, relatively new type of attack on the availability of Internet services and resources.[3] DDoS attacks are highly distributed, well-coordinated, offensive assaults on services, hosts, and infrastructure of the Internet. Effective defensive countermeasures to DDoS attacks will require equally sophisticated, well-coordinated, monitoring, analysis, and response.[4]

A malicious host controls large number of zombies which causes network congestion due to DDoS attack. Congestion control algorithm RED is used for congestion management. The RED algorithm is a representative AQM algorithm, and is also the only candidate algorithm recommended by RFC2309. The ability of AQM to detect incipient congestion and convey congestion notification to the end-hosts enables the sources to reduce their sending rates prior to buffer overflow. ECN is used in conjunction with AQM for signaling congestion to sources using packet marking instead of dropping packets.[5]

The basic idea of RED congestion control mechanism is to estimate the probability of packet marking for the realization of early notification on the calculation of the average queue length.[6] RED gateways keep the average queue size low while allowing occasional bursts of packets in the queue.[7] There are still some drawbacks in RED algorithm. Some improved RED algorithms are such as ARED, FRED, SRED and etc.[7] It is little bit difficult to set configuration parameters of RED to keep network environment stable. Theoretical analysis and simulation results all show that the packet loss and throughput are better that RED and ARED.

III. BASIC RED ALGORITHM

A. Random Early Detection Algorithm

RED can detect congestion by monitoring the average queue length of the output of router, and randomly chooses connections to notify congestion once the average queue length is close to congestion. The core of RED is to calculate the average queue length from the current queue length by the EWMA (Exponentially Weighted Moving Average)[6].The average length of the queue is calculate

$$avg = (1 - w_q) \bullet avg_q + w_q \bullet q \qquad \text{---(1)}$$

Where w_q is the queue weight of the instantaneous queue size, $0 \leq w_q \leq 1$; and during sampling, q is the instantaneous queue size. The actual queue size increases rapidly, for Internet traffics are burst or short-time congestion. The formula for temporary packet discard probability of RED is expressed as : [6]

$$\boldsymbol{p_b = \begin{cases} 0, & avg_q < min_{th} \\ 1 & avg_q < min_{th} \\ \frac{avg_q - min_{th}}{max_{th} - min_{th}} \bullet max_p & min_{th} \le avg_q \le max_{th} \end{cases}}$$ -

(2)

P_a : Current packet marking probability

$P_a = P_b/(1 - c \times P_b)$

where c is a counter to record the un-dropped (or unmarked) packets which has arrived after last dropped (or marked) packet. As the c increases, the final drop probability P increases slowly. So the dropping packets consecutively can be avoided.

Where max_p is the largest packet drop probability. Formula (2) shows the packet discard probability that depends upon the value of the average queue length (avg_q). Following Figure 1 shows the RED drop function.

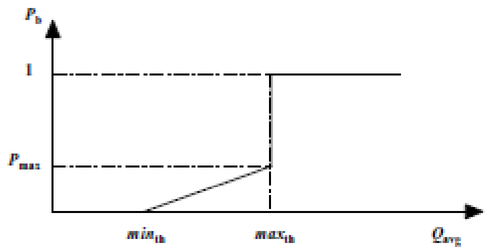


Figure.1.RED drop function

It is clear that RED algorithm monitors the average queue size and drops (or marks when used in conjunction with ECN) packets based on a set of statistical variables. And the calculation of drop rate for RED algorithm is related to the parameters of max_{th}, min_{th} and w_q as they mentioned above. RED performance is sensitive to the number of competing sources/flows. It is highly sensitive to its parameter settings. In RED, at least 4 parameters, namely, maximum threshold (max_{th}), minimum threshold (min_{th}), maximum packet dropping probability (P_{max}), and weighting factor (w_q), have to be properly set. RED performance is sensitive to the packet size. With RED, wild queue oscillation is observed when the traffic load changes.With the aim to get a more reasonable, dynamic and real time algorithm, this paper proposes BO-AURED which combines parameter settings of the RED algorithm with the buffer occupation and introduced upper threshold (U_{th}) to drop the packet less.

IV. PROPOSED BOAURED ALGORITHM

The motivation of BO-AURED is considering the variation of current buffer occupation effect and we have introduced new Threshold U_{th}(Upper threshold) for better use of buffer space, to queue more packets which reduces packet drops due to constant packet drop probability p_a is 1 when average queue size is greater than max_{th}. As in RED and other enhanced RED algorithm p_a increases linearly up to packet dropping probability max_p. If average queue size goes greater than max_{th} then p_a is set to 1 and all incoming packets are dropped. In order to get full advantage of the queue buffer packet drop-probability is calculated by another linear function when average queue size reaches between max_{th} threshold and U_{th} threshold. This paper combines the w_q, min_{th}, max_{th} ,maxp and U_{th} threshold parameters calculation with the current buffer occupations follows. In this paper, current buffer occupation is the rate that all of the current packets occupy the buffer.

$B_r = Q_c/B_f$ ----(3)

where, Q_c is instantaneous queue size; B_f is the setting of buffer size. And B_r is current buffer occupation in this paper. Though Floyd suggests us that w_q should be set as 0.002,it is hard to meet the requirements for low or heavy loads variation of network with static variable. When the value of itis not given, the NS2.35 uses (4) to compute the value of w_q,

$w_q = 1.0 - exp(-1.0/B_w)$ ------(4)

where B_w is the value of band width. In order to combine the w_q with current buffer occupation,
the algorithm replaces (4) with follow formula:

$w_q = [1.0 - exp(-1.0/B_w)] \times (1/B_r)$------ (5)

When the buffer occupation is low, the w_q should be enhanced. The algorithm will let more packets coming in the buffer, for this paper has changed the w_q. When the buffer rate is increased, the value of w_q should be turn down, for the buffer does not have large space to store lots of packets. When the buffer occupation is low, the w_q should be enhanced. The algorithm will let more packets coming in the buffer; for this paper has change the w_q. When the buffer rate is increased, the value of w_q should be turn down, for the buffer does not have large space to store lots of packets. The control results of RED are quite sensitive to the setting parameters of RED algorithm. Current rule of thumb is to set max_{th} to three times of min_{th} by Folyd suggestions. The setting of min_{th} is really important, it effect on both low average delay and high link utilization. But only the proper setting could get the balance of low average delay and high link utilization. Although the setting of min_{th} and max_{th} are quite important, Floyd still suggest us set min_{th} as a static, when it is been arranged in the simulation software. And the default value of min_{th} is 5 packets in NS2. During the simulation processes, operators always set the max_{th} half of the buffer size, and give the value of min_{th} one-third of the max_{th}. It is clearly that all these ways are setting the min_{th} and max_{th} as static value. They are not efficient ways to get perfect results for network congestion control, the network loads change all the time. Referring to Ref. [8], this paper takes instantaneous buffer occupation into consideration. According to the common used value of min_{th} and lots of experiments, min_{th} could be calculated as follows

$min_{th} = max(Q_{target},min)$-------(5)

where min_{th} takes the maximum value of Q_{target} and min.

$$min = \begin{cases} \frac{0.6 \times B_f - B_r \times 100}{3}; & (B_r < 0.6, 42 < B_f < 100) \\ \frac{0.6 \times B_f - B_r \times B_f}{3}; & (B_r < 0.6, B_f \ge 100) \end{cases} \text{----(6)}$$

This paper only takes the positive integer results from (6) for min_{th}. When B_f is smaller than 100 packets, to get the solutions of inequality, B_f should be larger than 42 packets. The algorithm also sets max_{th} to 3 times of min_{th}, so sets the denominator as 3 in the Eq. (6). Based on a large amount of experiments, we find the 0.6 which is the coefficient of B_f, is the best value to balance the drop rate and the delay time. When B_r is larger than 0.6.

$Q_{target} = targetdelay \times [B_w/(8 \times mean_{packet})]/2$------ (7)

where Q_{target} is the result variable of (5). The variable of *targe tdelay* will be set as target delay time in the simulation.

And $mean_{packet}$ is the value of average packet sizes. When the value of B_r is approximate to 0.6, to avoid min_{th} becoming too low, (5) checks the value of min_{th} real time. When the value of min is lower than the Q_{target}, then min_{th}

takes the value of Q_{target}.and when the value of *min* is higher than the Q_{target}, then min_{th} takes the value of *min.* U_{th} will be 4* of the min_{th} will be is set and U_{th} < ¾ th of the Buffer Size according to rule of thumb.[8]

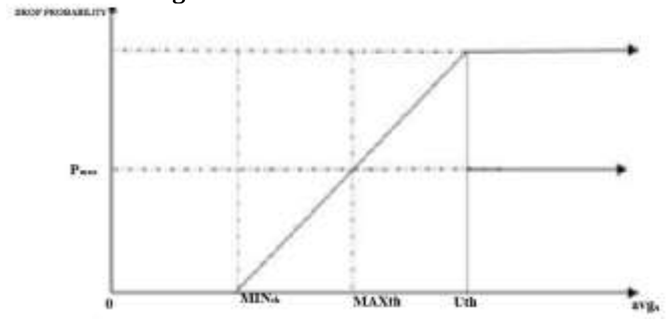
When average queue size is between max_{th} and U_{th} packet drop probability using our derived equation and packet is marked with probability p_b.

$P_b = (1 - Max_p)*((avg – Max_{th})/(U_{th} – Max_{th}))$---- (8)

$P_a = P_b(1- count *P_b)$------(9)

In the existing RED algorithm packet marking probability is directly set to 1 when average size reaches to the max_{th} so all the incoming packets are marked and dropped with probability 1 hence buffer space is wasted. For better buffer space utilization we have introduced new threshold U_{th}. In our proposed algorithm buffer space is utilized effectively, this will improve the performance of RED.

Following Figure: 2 shows the packet drop function of BO-AURED algorithm



FIGH.2. packet drop function of BO-AURED algorithm

There is another important parameter, *Pmax* to adapt the performance of algorithm. We know ARED's principal is to focus on adapting P_{max}, according to the changes in the network load, thus gives more stable queue size which means a predictable queue delay. ARED algorithm as shown in Fig.3

```
Every interval seconds:
If(Qavg>target and Pmax <0.5)
Increase Pmax
Pmax=Pmax+( α );
Else if (Qavg<target and Pmax >0.01)
decrease Pmax
Pmax=Pmax×( β );
```

Figh.4. adapting Pmax in ARED

Calculates P_{max} every interval seconds. Here, *target* is used as
a special target queue length, which is calculated as follows:

$target = 0.4 \times (max_{th} - min_{th})$-------(10)

In order to make P_{max} adapted network loads, this paper improve the method of calculating P_{max}. When buffer occupation is low, that means buffer has large space for the incoming packets, P_{max} should be decreased. When the buffer occupation comes high, P_{max} should be enlarged. The procedure of calculating P_{max} for BO-AURED is shown in Fig. 4. Adapting Pmax in BOAURED.

```
Every interval seconds:
If(Qavg>target and Pmax <0.5)
Increase Pmax
Pmax=Pmax+( α× Br );
Else if (Qavg<target and Pmax >0.01)
decrease Pmax
Pmax=Pmax×( β × Br );
```

Figh.4. adapting Pmax in BOAURED

V. SIMULATION AND RESULTS

As it is mentioned above, BO-AURED algorithm adapts to network variation automatically and real time. This paper performs a set of experiments using the NS2.35 The simulations aims to prove: first, BO-AURED algorithm has the smoother average queue; second, BO-AURED algorithm could get both low drop rate and low average delay time in the busty-traffic; third, when it comes to the heavy loads network environment, BO-AURED could adapt its parameters automatically and get low drop rate and test correctness of (5), especially sets the buffer size as 700 packets.

A. Network topology

Source ports of S_1 ... S_{n-1}, and S_n send TCP or UDP flows to destination port of D_1. Here n is a parameter, means source port number. The common settings of the three experiments are shown as follows: Source ports of S_1 to S_n send TCP or UDP flows to destination ports D_1; R_1 and R_2 are two routers; making R_1 loaded RED, ARED and BO-AURED respectively. The time of the four ports to begin is different by 1ms, 4ms, 8ms and 5ms [9].Parameter settings for RED and ARED: $w_q = 0.002$, $P_{max} = 0.02$. This paper sets $\alpha = 0.01$ and $\beta = 0.9$ to ARED in 3 experiments [10]. To BO-AURED, we no need to set any parameter rather buffer space, for they could adapt it parameters dynamically. Meanwhile, average packet sizes are 1000 bytes; also the target delay time is 5 ms for every algorithm.

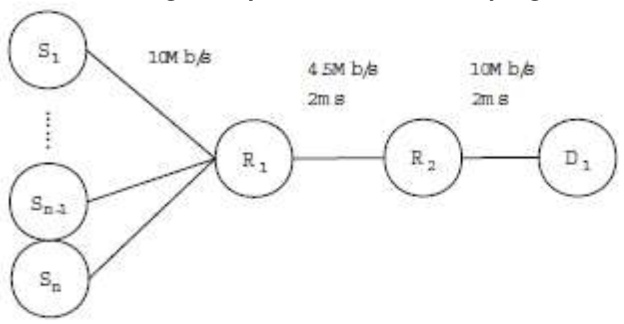


Fig.5. simulation topology

B. Experiment 1:testing the instantaneous performance
This experiment tests the instantaneous performance for RED,ARED,BO-AURED in 7-second simulation.

This paper also displays average drop rate, the number of arrived packets (short by arrived) and average delay time of 3 algorithms in Table I. The plots demonstrate that average queue length of BOAURED works more stable and smooth, especially at the beginning of the simulation with low average delay time. These experiments prove that, BOAURED algorithm has smooth average queue, lower drop rate and lower average delay time compared with other algorithms.

TABLE .I.TESTING THE INSTANTANEOUS PERFORMANCE

	drop rate (%)	arrived (packet)	average delay (s)
RED	0.8216	3774	0.0664
ARED	0.9801	3774	0.0608
BOAURED	1.1652	3774	0.0573

C. Experiment 2: testing the performance under the burst data flows

Experiment 2 illustrates the performance under burst. Table II also proof BOARED gets both low drop rate and low average delay time, for it binds the calculation of parameters dynamically. Because BOARED could allowed more packets to arrive in the buffer, when it hasn't been fully used. So it may made more packets arrive at the destination with a low drop rate and lower average delay time, when the network load becomes heavily.

TABLE .II.TESTING THE PERFORMANCE UNDER THE BURST DATA FLOWS

	drop rate (%)	arrived (packet)	average delay (s)
RED	1.9701	13548	0.0659
ARED	1.9500	13543	0.0582
BOAURED	1.8941	13543	0.0550

D. Experiment 3: testing the performance under heavy loads

Experiment 3 aims to test the performance under heavy loads for 3 algorithms. the BO-ARED becomes stable quickly and its average queue length plots are more smooth than other RED's. The data of experiment 3 is shown in Table III. It demonstrates that the BOAURED algorithm still takes the low drop rate with low average delay time under heavy loads. Experiment 3 proves that BOAURED has good performance under the high pressed the network environment.

TABLE.III.TESTING THE PERFORMANCE UNDER HEAVY LOAD

	drop rate (%)	arrived (packet)	average delay (s)
RED	4.988	27161	0.6597
ARED	10.71	27159	0.3831
BOAURED	5.53	27160	0.5639

It is clearly that BOAURED could achieve the lowest drop rate both during the short time simulations and long-time busty traffic simulations compared with RED, ARED and BOAURED, and also with low average delay time.

V. CONCLUSION

This paper has proposed a new algorithm BOAURED, which revises the sensitivity of parameter settings of RED. And it adjusts its parameters to adapt to network changing automatically and real time. The analysis and simulations all demonstrate that the BOAURED algorithm can be suitable to network variation rapidly and reduces both drop rate and delay time.

REFERENCES

1) Ketki Arora, Krishan Kumar, Monika Sachdeva "Impact Analysis of Recent DDoS Attacks" International Journal on Computer Science and Engineering (IJCSE)ISSN : 0975-3397 Vol. 3 No. 2 Feb 2011

2)Takanori Komatsu and Akira Namatame,"Effectiveness of close-loop congestion controls for DDoS attacks",Intelligent and Evolutionary Systems, Springer, Vol. 187, pp. 79-90, 2009.

3)Chandni M Patel, Viral H Borisagar," Survey On Taxonomy Of DDoS Attacks With Impact And Mitigation Techniques" International Journal of Engineering Research & Technology (IJERT) Vol. 1 Issue 9, November- 2012 ISSN: 2278-0181

4) Christos Papadopoulos, Robert Lindell, John Mehringer, Alefiya Hussain, Ramesh Govindan" COSSACK: Coordinated Suppression of Simultaneous Attacks" Proceedings of the DARPA Information Survivability Conference and Exposition (DISCEX'03) 0-7695-1897-4/03 © 2003 IEEE

5)Saurabh Sarkar, Geeta Sikka, Ashish Kumar,"Evolution and Optimization of Active Queue Management Algorithms over High Bandwidth Aggregates",International Journal of Computer Applications (0975 – 888) Volume 48– No.12, 11-16,June 2012 new RED

6) Dashun Que, Zhixiang Chen, Bi Chen,"An Improvement Algorithm Based on RED and Its Performance Analysis",ICSP2008 Proceedings, 978-1-4244-2179-4/08 2005-2008 ©2008 IEEE

7) S. Floyd and V. Jacobson, Random Early Detection Gateways for Congestion Avoidance,IEEE/ACM Transactions on Networking, Vol. 1, No.4, pages 397- 413, August 1993

8) S. Floyd. RED: Discussions of setting parameters. http://www.icir.org/floyd/REDparameters.txt

9) Xiaoping Yang, Hong Chen, Pingping Xiao, "An Algorithm of Enhancing RED Fairness", *The 7th World Congress On Intelligent Control And Automation, WCICA 2008*, pp.2149-2152, 1109/WCICA, Jun. 2008.

10) S. Floyd, Ramakrishna Gummadi and Scott Shenker, "Adaptive RED: An Algorithm for increasing the Robustness of RED's Active Queue Management.",[Online] available :nhttp://icir.org/floyd/papers/adaptiveRed.pdf, Aug. 2001.