BIO: An Alternative to RIO

Bahri Okuroglu and Sema Oktug

Department of Computer Engineering, Istanbul Technical University, Maslak, Istanbul, Turkey

ABSTRACT

RED (Random Early Detection) is the most popular active queue management algorithm, although it has some weaknesses. Recently, another active queue management algorithm, BLUE, was proposed and shown that it is more successful in controlling the queue length when high number of flows are active on ECN (Explicit Congestion Notification) capable networks. In this paper, RED and BLUE algorithms are evaluated for different levels of RTTs, with/without ECN support. It is shown that BLUE on ECN incapable networks is not as successful as on ECN capable networks.

Differentiated Services architecture suggests that RIO (Red with In and Out) style queue management algorithms are to be used on each AF (Assured Forwarding) queue to offer different levels of services for different priorities at each AF class. Inspired of BLUE's success over RED on ECN capable networks, we developed a simple alternative to RIO, BIO (BLUE with In and Out). BIO, which runs two different BLUE algorithms for in and out packets, was expected to achieve lower loss rates while maximizing link utilization for high number of active flows on AF queues.

However, due to the self-configuring architecture of the algorithm, it is observed that BIO marks packets too aggressively and degrades utilization. In this paper, the properties of BIO are also explained and the results obtained are justified.

Keywords: Active Queue Management, RED, BLUE, RIO, Quality of Service, Differentiated Services, TCP/IP

1. INTRODUCTION

The benefits of the stateless architecture of the IP (Internet Protocol) have enabled the rapid growth of the Internet. With this enormous growth, network congestion, caused by the stateless architecture of IP has become more apparent. As the achievement of network efficiency and the reduction of the loss rate became major problems, new mechanisms are required to meet the expectations of today's applications since the architecture of the Internet is not designed to support these kinds of applications.

Network congestion may lead to total failure of the network when proper congestion control algorithms are not used. This failure is named as congestion collapse and predicted by Nagle in 1984 [1]. Jacobson's congestion control algorithms [2], which had been developed upon the first observations of Nagle's predictions on a real network, are still being used on the current Internet.

Jacobson proposes slow start and congestion avoidance algorithms to be run by the end-node TCPs to prevent network congestion. Slow start algorithm increases the data in transit to start the self-clocking [2]. Congestion avoidance algorithm ensures that the end-node takes the required action on packet loss, which means there has been congestion on network.

Congestion grows exponentially and early detection of the congestion helps to prevent it. This decreases the drop rate caused by the congestion. Gateways experiencing the congestion are the only nodes, which can detect the congestion and take proper action earliest.

Drop-tail queues, which employ the traditional queue mechanism, do not perform any special processing on the queue for congestion control. Packets are accepted while the queue length is less than a pre-defined limit, and all the packets are dropped after this limit. Several problems are inherent in this architecture as the queue becomes full almost all the time.

Active queue management mechanisms are congestion control algorithms, which are run on the gateways to detect congestion earlier and to send implicit or explicit feedback to the end-nodes. Due to the advantages, the use of the active queue management architectures on the gateways is recommended by IETF [3]. These algorithms have many advantages

This work is financially supported by Istanbul Technical University Research Fund under grand number 1076.

Correspondence: Bahri Okuroğlu, okuroglu@itu.edu.tr; Sema Oktuğ, oktug@itu.edu.tr

over classical drop-tail queues. Although the main advantage is the detection of the congestion earlier and reduction in the drop rate, these mechanisms also avoid the global synchronization and bias against bursty flows.

Active queue management algorithms either drop or just mark packets in order to control the queue length. When packets are dropped to control congestion, standard TCP congestion control algorithms are efficient for the end-nodes. For the marking case, ECN-capable TCP must be used. To prevent high loss rates caused by the packet drops to control the queue length, IETF is considering deployment of ECN [4]. By the use of ECN, gateways can mark packets instead of dropping them, and end nodes can take the proper action to decrease the size of the congestion window which indicates how much data the network can handle and is managed by the end node TCP.

New kinds of applications, born with the growth of the Internet, need new kind of services over Internet. IETF has been developing Differentiated Services (DiffServ, DS) architecture [5] to offer different levels of services to the applications beyond the best-effort service. Scheduling algorithms at gateways, which serve to different queues, are used to support this new architecture, which is more scalable than the previous developments in this area. Besides the drop-tail and active queue mechanisms, new queue management algorithms, which are developed especially for service differentiation, are used on these queues. These new algorithms are based on the existing queue management algorithms and offer service differentiation on a single queue.

The rest of the paper is organized as follows: First, the most popular active queue management algorithms RED and BLUE will be investigated using the simulations. Next, DS architecture and related queue management algorithm, RIO, are explained briefly. Finally, a new queue management algorithm is proposed and discussed for the DS nodes.

2. ACTIVE QUEUE MANAGEMENT ALGORITHMS

RED [6] is the most popular and widely used active queue management algorithm. Recently an alternative algorithm called BLUE [7] is developed. In this section these two active queue management algorithms are briefly described.

2.1. RED

RED, a development on the previous works like ERD (Early Random Drop) [8] and DECbit [9], is the first widely used active queue management algorithm. ERD drops each arriving packet at the gateway with a fixed drop probability once the queue length exceeds a certain drop level. It is shown that misbehaving users receives higher throughput with ERD [10]. DECbit is a binary feedback scheme for congestion control. The average queue length is calculated using the last two periods upon a packet arrival. The congestion indication bit is set if the average length exceeds one.

RED differs from ERD in two ways: First RED can also mark packets instead of dropping. Second, RED's packet marking probability is a function of the average queue length, not the instant queue length. The differences between the DECbit and RED are the calculation of the average queue length, marking probability and the method of sending feedback to the end nodes. While DECbit uses last two periods for the average queue length calculation, RED uses the full history of the queue length. RED also uses a randomized algorithm for marking packets. The last difference between these algorithms is their methods to send feedbacks. While RED can both mark and drop packets, DECbit can only mark packets. With this method, RED can work even without the cooperating end nodes.

```
for each incoming packet
    calculate the average queue length (avg)
    if $min_{th}<avg<max_{th}$
        calculate packet dropping probability $p_a$
        mark packets with probability $p_a$
    if $max_{th} \leq avg$
        mark packet

($avg \leftarrow (1 - w_q) * avg + w_q * q$)
```

Figure 1. RED Algorithm

RED algorithm given in Figure 1 calculates the average queue length using both the actual queue length and the average queue length, and *marks* packets with a probability proportional to the average queue length. w_q is the first parameter of RED which is used as weight of the average queue length in calculation of the new average queue length. By using average queue length at the marking probability calculations, instant bursts can be tolerated. Marking packets provides feedback information to source nodes on the congestion level of the gateways through the path.

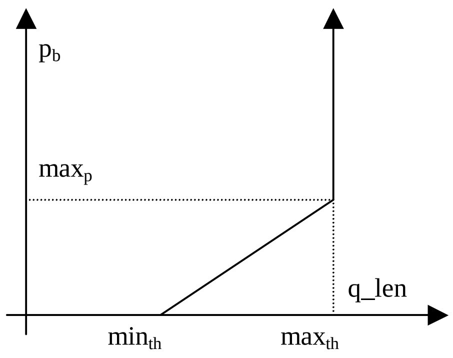

Figure 2. RED parameters

The primary design goal of RED is to avoid congestion by controlling the average queue length. Additional goals are avoiding global synchronization and bias against bursty traffic. RED uses two parameters, min_{th} and max_{th}, on the queue length that show threshold values. Once the queue length exceeds the min_{th} parameter, packets are marked with the randomized marking probability. max_p is the maximum packet marking probability. Last parameter, shown in Figure 2, is the queue weight in the calculation of the average queue length using the below formula:

$$avg \leftarrow (1-w_q)*avg + w_q*q \qquad (1)$$

The behavior of the algorithm can be quite different for different set of parameters. These parameters should be tuned considering the network and traffic architecture.

The effectiveness of the RED algorithm on congestion control and reduction of the loss rate is proven by both simulations and real-world experiences [6, 11]. However, the algorithm has parameter selection problems [12**,** 13]. In order to benefit from the algorithm its parameters must be arranged properly. Moreover, the marking aggressiveness of the algorithm is insensitive to the number of active flows on the gateway. When the bottleneck link is shared equally between the active flows, marking one packet to send congestion feedback decreases the total throughput by the rate of $1-1/(2N)$ [14]. Since only the average queue length and max_p parameter are used in calculation of RED's packet marking probability, RED cannot mark packets proportionally to the number of active flows. This may result in the loss of queue control.

Feng proposed a modification to the RED algorithm in order to adjust its parameters dynamically according to the load on the gateway in [14]. With this modification, marking probability is increased by factor of β upon queue overflow event, and decreased by factor of α upon queue idle event. Feng verified the effectiveness of this algorithm by simulations. When large numbers of flows are active at gateways, RED causes high loss rates and inefficient use of link capacity even with the use of ECN and adaptive parameter adjustment mechanisms [7].

2.2. BLUE

BLUE [7] is a recently developed active queue management mechanism which has a completely different marking strategy compared to RED.

BLUE algorithm given in Figure 3, assumes that queue length does not directly reflect the congestion level. Hence it does not update the packet marking probability with the queue length. Instead it uses queue overflow and idle event history to update the packet marking probability (p_m). Packet loss due to the queue overflow means that the marking is not aggressive enough and p_m should be increased. Similarly, the queue idle event occurs as a result of the aggressive marking policy therefore the p_m parameter should be decreased. This mechanism effectively allows BLUE to *learn* the correct rate it needs to send back congestion notification.

```
For each packet loss:
    if ((now – last_update) > freeze_time )
            $p_m = p_m + d_i$;
            last_update = now;
For link idle event:
    if ((now – last_update) > freeze_time )
            $p_m = p_m - d_d$;
            last_update = now;
```

Figure 3. The BLUE Algorithm

BLUE uses three parameters: The first two parameters determine the amount by which p_m is incremented in case of the queue overflows (d_i) or is decremented when the link is idle (d_d). The last parameter is the minimum time interval between two successive updates (*freeze_time*). BLUE is a simple algorithm compared to RED.

3. PERFORMANCE COMPARISON

Performance of the algorithms is evaluated by simulations and an experimental testbed at [7]. Even it was shown that BLUE beats RED when ECN is used in [7], BLUE has not been evaluated for different levels of RTTs and without ECN support. In this paper, performance of the BLUE algorithm and the RED algorithm is evaluated and compared by simulations using ns simulation tool [15], which is also used in several other papers and RFCs. The simulations are repeated for different RTTs considering networks with and without ECN support.

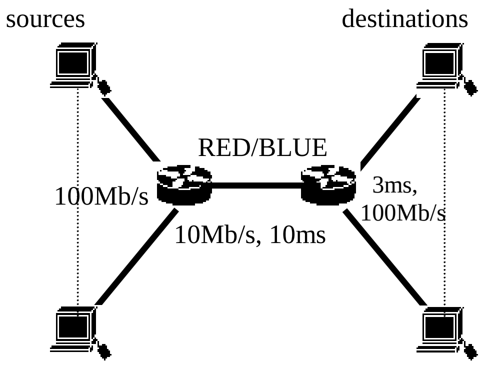

Figure 4. Simulation network architecture

The network architecture used in the simulations is shown in Figure 4. Twenty source nodes are connected through a bottleneck link to the twenty destination nodes. Each node runs fifty simultaneous exponential traffic applications with 2ms average ON, 3ms average OFF times with 1000 bytes packets. All the nodes are connected to the 10Mb/s, 10ms bottleneck with 100Mb/s links. The delays of the links between the sources and the bottleneck are selected differently for each link to differentiate the RTTs of flows. Flows are started randomly in the first 70 seconds of a 200-second simulation and the measurements are taken during the second half of the simulation duration to let BLUE have enough time to learn the traffic characteristics. All nodes employ TCP Reno.

Either a RED or a BLUE queue is used at the bottleneck link. Simulations are repeated for various queue sizes. RED queue is configured as $min_{th} = q\_size/4$ and $max_{th} = 3min_{th}$ where *q_size* is the queue capacity. The marking limit of RED, max_p, is selected as 1 which is the best selection for this quantity of flows [7]. w_q parameter is selected as 0.002. BLUE queue is configured as $d_i = 0.0005$, $d_d = 0.001$ and *freeze_time* = 10ms. These BLUE parameters are the optimized values to increase the throughput and decrease the loss rate [16].

Performance of the RED and the BLUE algorithms is evaluated for two different levels of RTTs, with and without ECN support. First, in Case 1, the large RTT, ECN capable network environment is used. Next, in Case 2, the RTTs of the flows are decreased while keeping ECN support. Finally, in Case 3, large RTT, and ECN incapable network environment is used.

3.1. Large RTT flows with ECN support

In this case, the link delays between the sources and the bottleneck are selected as (2*i*+200)ms where *i* is the source node index between 0 and 19. All the nodes are ECN capable and gateways mark the packets instead of dropping them. Figure 5 reveals that BLUE performs better than RED for each queue size, when throughput is considered. BLUE converges to the 100% link utilization using small queues. However, RED can reach that efficiency only for large queues. The inefficient link usage of RED is caused by its inability to control the queue length for high number of active bursty flows. As it is seen in Figure 6, both the instant queue length and the average queue length of RED oscillate between the queue limits. As a result, the queue overflows just after a link idle event and becomes empty just after a queue overflow.

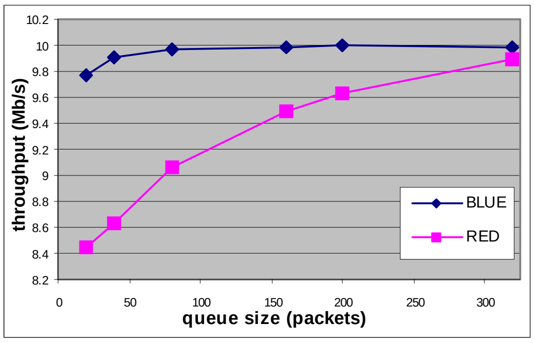

Figure 5. Total throughput over bottleneck link for both RED and BLUE (large RTT)

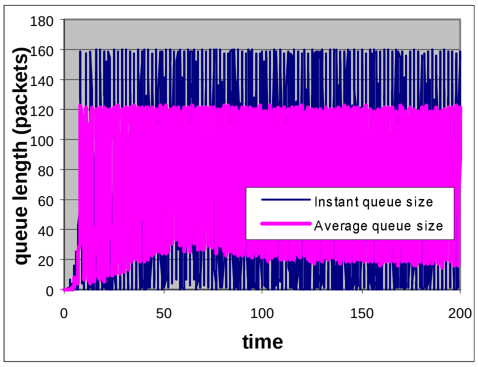

Figure 6. RED average and instant queue size graph for q_max = 160 (large RTT)

Figure 8 shows that BLUE can control the queue successfully over time once the behavior of the traffic is learned. When new sources are activated, BLUE suffers from the changing traffic load. Once the traffic is stabilized, BLUE can control the queue successfully since it *knows* how to mark the packets. This control keeps the link at the high througput condition.

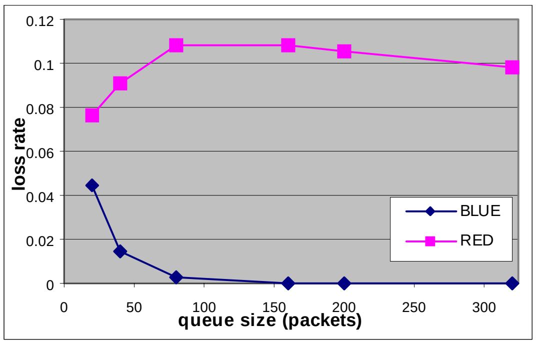

Figure 7. Total drop rate over bottleneck link for both RED and BLUE (large RTT)

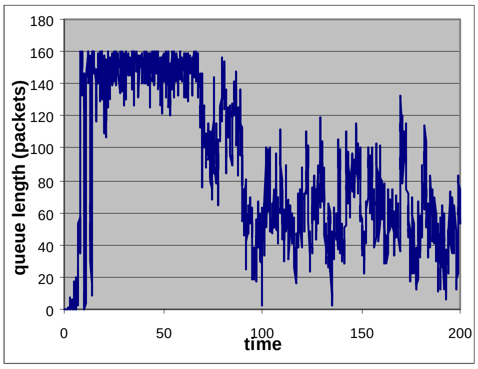

Figure 8. Queue size for BLUE when q_max = 160 (large RTT)

In Figure 7, drop rates obtained for RED and BLUE can be compared. The queue instability occurring in RED causes large drop rates besides low link utilization. These drops are mainly forced drops, which are caused by queue overflows. However, as shown in Figure 7, when queue size is increased BLUE converges to zero drop rate. Since no queue overflow is occurred and ECN is used to mark packets, packet loss can be kept at minimum rate.

3.2. Small RTT flows with ECN support

Here, the Case 1 environment is modified by decreasing RTTs. The link delays of the sources to the bottleneck are set to (2*i*+5)ms where *i* is the node index between 0 and 19.

Figure 9 shows that RED performs better under this configuration when throughput is concerned. The reason for such a result is again RED's inability to control the queue successfully. As seen in Figure 10, RED behaves like a drop-tail queue and causes too much forced drops to appear under this configuration. Since the queue is full most of the time, the throughput is much better than the one in the previous case. However, this means benefits of active queue management algorithms cannot be observed.

The throughput of BLUE is close to the results obtained in the previous case. As it is shown, RED and BLUE cannot beat each other on the throughput basis when queue size greater than 100 packets.

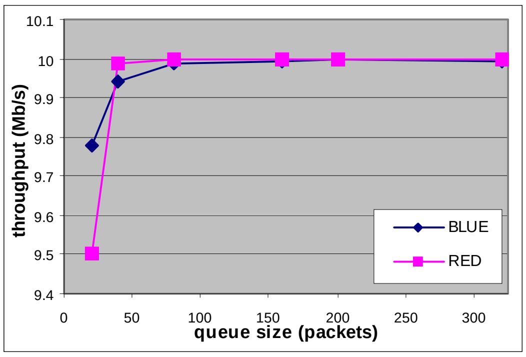

Figure 9. Total throughput over bottleneck link for both RED and BLUE (small RTT)

The most important observation of this simulation case is that RED cannot control the queue even it marks packets at its maximum rate, which is 1. The marking rate calculation algorithm of RED, which uses queue length as the congestion measure, causes this.

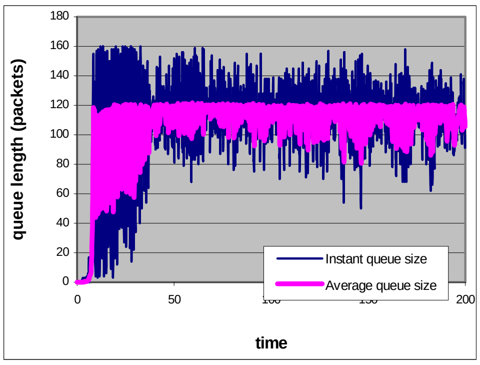

Figure 10. RED average and instant queue size graph for q_max = 160 packets (small RTT)

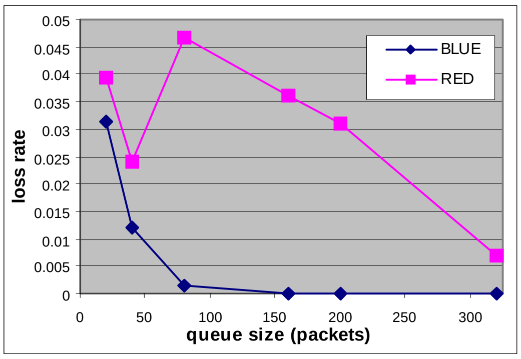

Figure 11. Total packet loss rate over bottleneck link for both RED and BLUE (small RTT)

Figure 10 shows that RED cannot control the queue and high number of forced drops caused by the queue overflow result in high loss rate as seen in Figure 11. However, BLUE can still control the queue successfully, it can keep the link at low loss rate.

The Case 1 and Case 2 simulations show that BLUE achieves queue control much better than RED in the case of high number of active flows with ECN support. For large RTTs, the RED queue overflows and idles many times. When the RTTs are decreased, the queue only overflows.

3.3. Large RTT flows without ECN support

In order to investigate the behavior of BLUE using the nodes which do not support ECN, the Case 1 configuration is re-run without ECN support.

As seen in Figure 12 and Figure 13, BLUE loses its superiority against RED observed in Case 1 and Case 2. The throughput values obtained for RED and BLUE are close to each other and the drop rate of BLUE is a bit higher than that of RED. Figure 14 shows that RED can control the queue successfully when large RTT is used and without ECN support. As a result, the overflow loss for RED decreases. Without the ECN support, the RED queue does not oscilate as in Case 1. Both

RED and BLUE suffer from high loss rates, but large portion of these losses are unforced drops, which are caused by active queue management mechanisms to control the queue.

BLUE behaves worse than it behaves in Case 1 on queue control as it is seen in Figure 15. Since it cannot use packet marking to send feedback, it also suffers from high loss rates as shown in Figure 13.

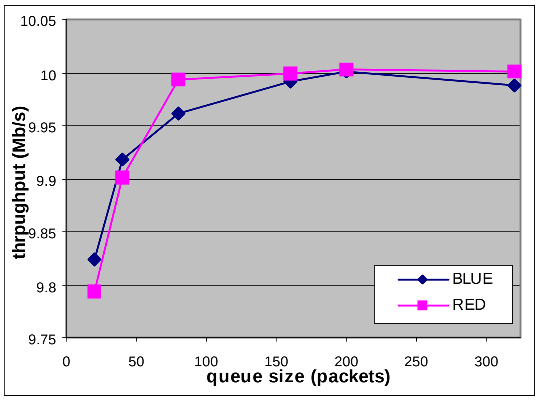

Figure 12. Total throughput over bottleneck link for both RED and BLUE (large RTT, without ECN)

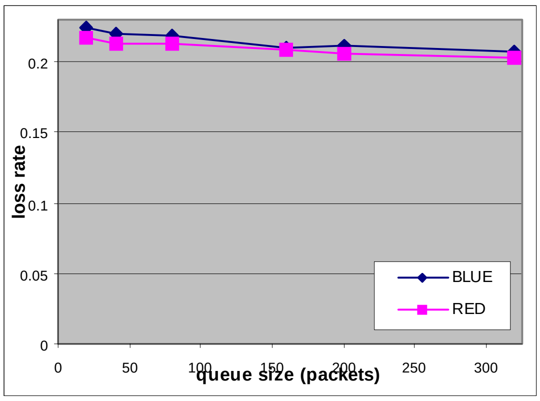

Figure 13. Total drop rate over bottleneck link for both RED and BLUE (large RTT, without ECN)

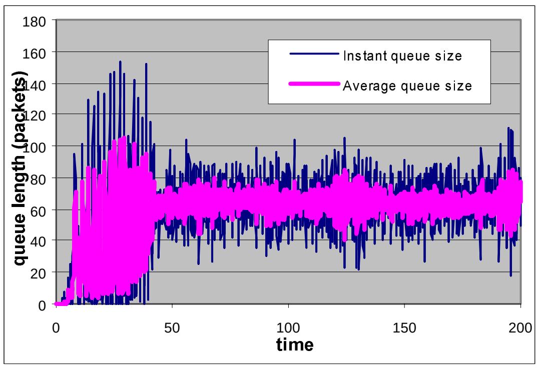

Figure 14. RED average and instant queue size graph for q_max = 160 packets (large RTT, without ECN)

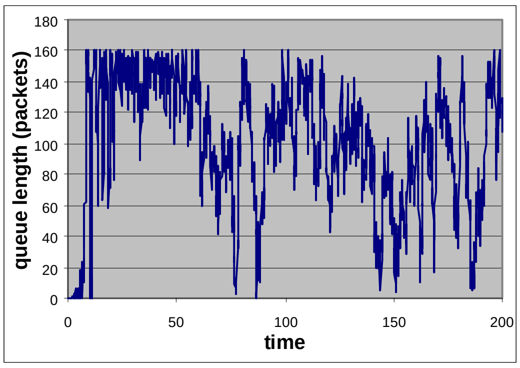

Figure 15. BLUE queue size graph for q_max = 160 (large RTT)

The major difference between the results of Case 1 and Case 3 simulations is the behavior of BLUE. The performance of BLUE degrades as ECN support disappears. However, the performance obtained for BLUE is still very close to that obtained for RED.

These simulations are also repeated for smaller RTTs and similar results are obtained.

4. DIFFERENTIATED SERVICES MECHANISMS ON ROUTERS

Differentiated Services (DS) is a new architecture [5], which is developed by IETF to support different levels of services over IP networks. It is more scalable than the previous solution, Integrated Services [17], which requires complex checks on each packet and maintenance of state information for each active flow on each gateway.

This architecture is based on DS domains, which is managed by a central authority. The gateways on the borders of this domain classify and mark the DS field of the packets to indicate the service level to be offered by the network. This decision is based on the Service Level Specification (SLS), which is agreed on by the neighbor network management. This classification is a multi-field (MF) classification which means that multiple fields of the packets like source and IP addresses, ports are used in classification.

Internal nodes perform behavior aggregate (BA) classification on packets, which means that only the DS fields of the packets are checked. Different forwarding treatments (PHB, Per Hop Behavior) are offered to the packets based on this classification. Most common method of service differentiation is using scheduling algorithms for different number of queues for each output interface.

DS provides Expedited Forwarding (EF) [18] PHB (Per-Hop Behavior) which offers “low loss, low latency, low jitter, assured bandwidth” end-to-end service and Assured Forwarding (AF) [19] PHB which provides delivery of IP packets in four independently forwarded AF classes. Within each AF class, an IP packet can be assigned one of three different levels of drop precedence. AF PHB should also tolerate short-term congestion.

4.1. Red with In and Out (RIO)

To offer this kind of services a router may use different number of queues for each output interface served with a scheduler. Gateway may have an EF queue, a BE queue and four AF queues. Each AF queue should run an algorithm like RIO (RED with In and Out) [20] to be able to serve packets with different priorities in one queue.

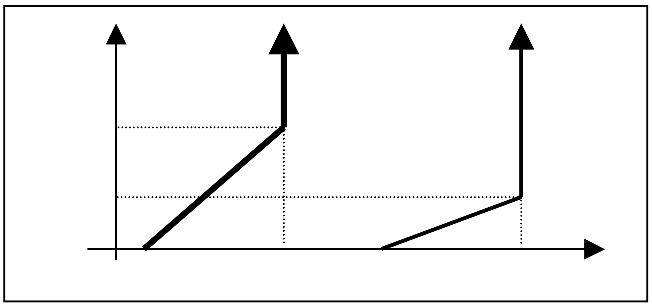

Figure 16. RIO parameters

As seen in Figure 16, RIO runs *N* (3) different parameterized RED algorithm on the queue to mark each packet respective to its priority. It begins to drop low priority packets much before the high priority packets. When the congestion increases high priority packets are dropped. [20] shows that RIO can provide different levels of services to different priority packets.

4.2. BLUE with In and Out (BIO)

Three different priority packets from one AF class can be served using a RIO queue to offer DS functionality. RIO provides different levels of services to each priority.

RIO is an algorithm based on RED. For high number of active flows on the gateway, RED loses the control of the queue. It is shown that BLUE performs better than RED under such a configuration. Inspired of BLUE’s success, we developed a simple alternative to RIO, named BIO (Blue with In and Out).

With the algorithm in Figure 17, two distinct BLUE algorithms are run for in and out packets. While total number of packets are compared with the out limits (max_{out} and min_{out}), only the in packet count is used in comparison with the in limits (min_{in} and max_{in}), similar to RIO.

Performance evaluation of the algorithm is tested on network given in Figure 4. Ten nodes are used with 100 active flows, which are transferring an infinite-length file. RIO and BIO queues are used for AF queues on router1. ECN is enabled on the network since BLUE is more successful with ECN support. Bottleneck link bandwidth is 70Mbps and all other links are 100Mbps. While the bottleneck link delay is 4ms, delay of links connecting to the bottleneck link is 15ms. Queue for bottleneck link can keep 100 packets of size 1000 bytes. Token-bucket conditioners are employed between source nodes and bottleneck link router. While the peak rate is set as 105Kbps for FTP agents on even nodes, it is set as 21Kbps for FTP agents on odd nodes to fill the 90% of the bottleneck link capacity with in packets. Token bucket capacity is set as the half of the peak rate for all FTP agents. RIO parameters are chosen as min_{out}=10, max_{out}=30, $maxp_{out}$=0.5, min_{in}=40, max_{out}=70, $maxp_{in}$=0.02. w_q is set to 0.002 as in [20], but it makes the RIO queue to oscillate widely. To prevent this behavior we set w_q as 0.004. $maxp_{in}$ is set as 0.02 as in [20], but $maxp_{out}$ is set to 0.5 to prevent packet losses caused by queue overflow. BIO parameters are chosen as $d_{i,in}$=0.0005, $d_{d,in}$=0.01, $d_{i,out}$=0.001, $d_{d,out}$=0.01, max_{in}=$qlim$, min_{in}=max_{in}/2,

max_{out}=min_{in} and min_{out}= max_{out}/3. Simulations are run for 200 ms and all the sources are randomly started in first 70 ms of the simulation. Bandwidth measurements are taken during the first 90-190 ms.

```
**On packet enque**
  if (packet is in profile)
    mark packet with $p_{m,in}$
  else
    mark packet with $p_{m,out}$
  if $Q_{len,in}$>$max_{in}$ and (now-last_in_inc) > in_hold
    $p_{m,in} = p_{m,in} + d_{i,in}$
    last_in_inc = now
  if $Q_{len}$>$max_{out}$ and (now-last_out_inc) > out_hold
    $p_{m,out} = p_{m,out} + d_{i,out}$
    last_out_inc = now
**On link ready for transmission event**
  if link is idle or $Q_{len,in}$<$min_{in}$
    if (now-last_in_dec) > in_hold
      $p_{m,in} = p_{m,in} - d_{i,in}$
      last_in_dec = now
  if link is idle or $Q_{len}$<$min_{out}$
    if (now-last_out_dec) > out_hold
      $p_{m,out} = p_{m,out} - d_{i,out}$
      last_out_dec = now
```

Figure 17. Two-level BIO algorithm

Figure 18 and Figure 19 show the queue lengths for BIO and RIO queues during the simulation. The main observation is that queues do not reach the queue limit, and are successfully controlled by RIO. RIO algorithm's success in controlling the queue against RED is a result of marking out packets early. It is seen that BIO cannot control the queue successfully and causes queue overflows.

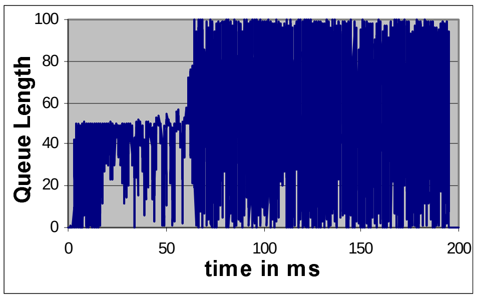

Figure 18. BIO queue length in packets

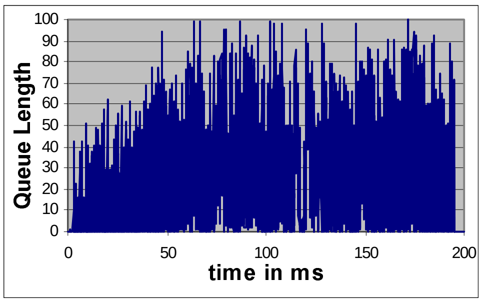

Figure 19. RIO queue length in packets

Initially length of both queues is too low, since there are not enough active flows to consume the bottleneck bandwidth. In the first 1/3 of the simulation, where flows are started, BIO queue length is 50 at maximum since there are too much out packets of the started flows in queue. BIO cannot *mark* these out packets initially and these out packets are dropped when the queue size reaches max_{out} . RIO queue can go beyond max_{out} since it uses average queue length in marking decisions.

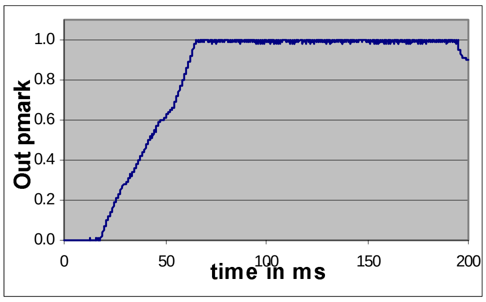

Figure 20. Out packet marking probability of BIO

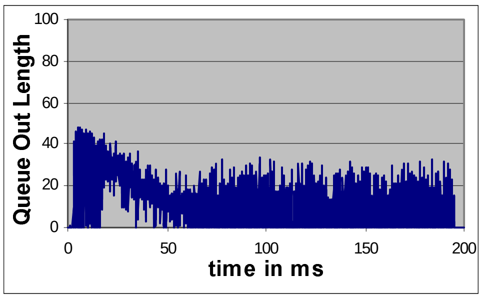

Figure 21 BIO queue out length

As seen in Figure 20, BIO queue's $p_{m,out}$ reaches to 1 and remains there till the end of the simulation. Total number of in and out packets in queue is used in $p_{m,out}$ update process. Since 90% of the link capacity is expected to be used by in packets and there always exist large number of packets in queue and $p_{m,out}$ is always increased. This aggressive marking probability for out packets causes small number (max of min_{out}) of out packets in the queue as seen in Figure 21.

Figure 22 shows the total bandwidth of flows on each node. It is seen that both queue management algorithms can reach the total in profile rate, and they cannot be said as beating each other. While the loss rate for RIO is measured as 0.1974, it is measured as 0.1796 for BIO. This high packet drop rate is caused by the inefficient link capacity for out packets.

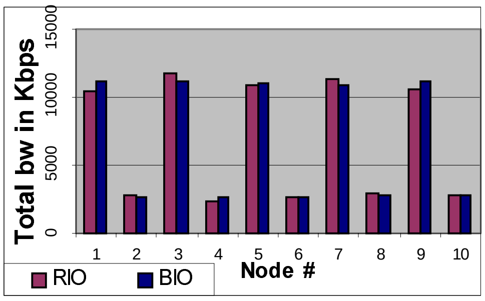

Figure 22 Total bandwidth of flows on each node

BIO's behavior in updating the out packet marking probability can be generalized to self-configuring 2-level queue management mechanisms if they are counting both in and out packets for the update process of out packet marking probability.

5. CONCLUSION

Active queue management algorithms detect congestion earlier and convey notification to sources by dropping or marking packets. These mechanisms increase throughput and decrease loss rate.

RED, the most popular active queue management algorithm, has inherent problems besides its proven benefits. A new active queue management algorithm, BLUE, can perform better with the use of ECN when high number of flows are active at the gateways.

In this paper, the performance of RED and the performance of BLUE are compared by changing RTT values and ECN support under heavy bursty traffic. It is shown that: When ECN, which helps in the reduction of the loss rate, is used with RED, it loses the queue control in case of large number of active flows. For some conditions RED behaves like a drop-tail queue, for others queue size oscillates between the queue limits. However, ECN supported BLUE algorithm can control queue successfully and decrease the loss rate for central gateways with high number of active flows. When ECN support disappears, the performance of BLUE becomes closer to that of RED.

Today ECN is not deployed widely, thus BLUE's benefits cannot be observed in real world. Even if BLUE cannot surpass the performance of RED without ECN support, it can still race with it, and for ECN capable hosts and networks it will help to decrease the packet loss rate.

Inspired of the success of the BLUE against RED for high number of flows, a new queuing algorithm, BIO, to be used in Differentiated Services nodes is proposed. BIO was expected to achieve lower loss-rate and higher link utilization. However, due to the two level packet discard algorithm and self-configuring architecture, BIO cannot be said as superior to RIO.

Currently, we are working on the parameter calibration of BIO in order to get better results.

REFERENCES

1. **Nagle, J.**, 1984, Congestion Control in TCP/ IP Internetworks, Request For Comments: 896

2. **Jacobson, V.**, 1988, Congestion Avoidance and Control, Proceedings of SIGCOMM ’88, Stanford, CA, ACM

3. **Braden, B., et. al.,** 1998, Recommendations on Queue Management and Congestion Avoidance in the Internet, Request For Comments: 2309

4. **Ramakrishnan, K., Floyd, S.**, 1999, A Proposal to add Explicit Congestion Notification (ECN) to IP, Request for Comments: 2481

5. **Blake, S., et.al., 1998**, An architecture for Differentiated Services, Request for Comments: 2475

6. **Floyd, S., Jacobson, V.**, 1993, Random Early Detection Gateways for Congestion Avoidance, IEEE/ACM Transactions on Networking

7. **Feng, W., Kandlur, D., Saha, D., Shin, K.**, 2000, BLUE: A new class of active queue management, http://www.eecs.umich.edu/~wuchang/blue/

8. **Jacobson, V**., Congestion Avoidance and Control, Proceedings of SIGCOMM ’88, August 1988, pp.314-329.

9. **Ramakrishnan, K.K., and Jain, R.**, 1990, A binary feedback scheme for congestion avoidance in computer networks, *ACM Transactions on Computer Systems*, V.8, N.2, pp. 152-181

10. **Zhang, L.**, A New Architecture for Packet Switching Network Protocols, MIT/LCS/TR-455, Laboratory for Computer Science, Massachusetts Institute of Technology, August 1989

11. **Doran, S.**, 1998, EBONE Interface Graphs, http://adm.ebone.net/~smd/red-1.html

12. **Labrador, M., Banerjee, S.**, 1999, Packet Dropping Policies for ATM and IP Networks, IEEE Communications Surveys, vol 2, no 3

13. **May. M., Bolot, J., Diot, C., Lyles, B.**, 1999, Reasons not to deploy RED, INRIA Sophia-Antipolis, ENSIM, Sprintlabs

14. **Feng, W,. Kandlur, D., Shin, K.**, 1999, A Self-configuring RED Gateway, INFOCOM ’99

15. NS: Network Simulator, http://www-mash.cs.berkeley.edu/ns/

16. **Okuroğlu, B., Oktuğ, S.**, 2000, TCP/IP’de Aktif Kuyruk Yönetim Mekanizmaları ve Internet İçin Kaliteli Hizmet, MS Thesis, İstanbul Technical University, Control and Computer Engineering

17. **Braden, R., Clark, D., Shenker, S.**, 1994, Integrated Services in the Internet Architecture: an Overview, Request for Comments: 1633

18. **Jacobson, V., Nichols, K., Poduri, K.**, 1999, An Expedited Forwarding PHB, Request for Comments: 2598

19. **Heinanen, J., Baker, F., Weiss, W., Wroclawski, J.**, 1999, Assured Forwarding PHB Group, Request for Comments: 2597

20. **Clark, D., Fang, W.**, Explicit Allocation of Best Effort Packet Delivery Service, http://diffserv.lcs.mit.edu/Papers/exp-alloc-ddc-wf.pdf