PERFORMANCE ANALYSIS OF ACTIVE QUEUE MANAGEMENT ALGORITHMS

Santhi V. [1], **Natarajan A.M.**[2]
[1]Lecturer, Department of Computer Science and Engineering,
PSG College of Technology, Coimbatore, India
[2]CEO, Bannari Amman Institute of Technology, Sathyamangalam, India.
Email: [1]sannthi@yahoo.com

Abstract

Active Queue Management (AQM) can potentially reduce packet loss rate in the Internet. This is used by routers for control congestion, where packets are dropped before queues become full. A number of active queue management algorithms for TCP/IP networks such as random early detection (RED), Fair RED (FRED), BLUE, Stochastic Fair Blue (SFB), and Core-stateless Fair Queuing (CSFQ) have been proposed in the past few years. This article presents a comparative study of these algorithms using ns-2 simulations. The performance metrics used in the study are queue size, packet drop marking probability, packet loss rate and bandwidth utilization. The study shows that BLUE is better than RED to avoid global synchronization for maintaining single marking probability. And also, the results shows that, among the five algorithms, SFB and CSFQ are more effective at stabilizing the queue size and controlling the packet loss rate among non-responsive flows while maintaining high link utilization. The performance of SFB and CSFQ are obviously better than that of RED, FRED and BLUE.

Key words : Congestion Control, Fair Queue, Networks, Queue Management

I.INTRODUCTION

Congestion in a network or internet creates obvious problems for the end system: reduced availability and throughput and lengthened response times. When a packet is dropped before it reaches its destination, all of the resources it has consumed in transit are wasted. Improving the congestion control and queue management algorithms in the Internet has been one of the most active areas of research. The Internet has mainly relied on the cooperative nature of TCP congestion control in order to limit packet loss and fairly share network resources. However new applications are being deployed which do not use TCP congestion control and are not responsive to the congestion control and are not responsive to the congestion signals given by the network. Such applications are potentially dangerous because they drive up the packet loss rates in the network and can eventually cause congestion collapse.

The aim of this paper to compare the performance analysis of active queue management techniques called Blue, Stochastic Fair Blue, and Core-stateless Fair Queuing with RED. Blue and Stochastic Fair Blue is used to reduce packet loss rates, queuing delay, high link utilization with minimal amount of buffer space. Core-stateless fair queuing enforces fairness among large number of connection with small amount of state information.

BLUE uses the packet loss and the link utilization history to manage congestion. Also, only a single marking probability is maintained, when the queue is continually dropping packets due to buffer overflow, this probability is incremented, and when the queue is idle or empty, it is decremented.

SFB, to detect the non-responsive flows and to protect the other TCP flows from such flows based on accounting mechanisms.

In CSFQ, edge routers compute per-flow rate estimates and label the packets passing through them by inserting these estimates into each packet header. Core routers use FIFO queuing and keep no per-flow state. They employ a probabilistic dropping algorithm that uses the information in the packet labels along with the router's own measurement of the aggregate traffic.

The rest of the paper is organized as follows. Section II gives a description of RED, FRED, SFQ and shows why it is ineffective at managing congestion. Section III describes BLUE, Stochastic Fair BLUE and Core-Stateless Fair Queuing and provides a detailed analysis. Section IV describes evaluation of its performances based on simulations. Finally, Section V concludes with a discussion of future work.

II. BACKGROUND

One of the biggest problems with TCP's congestion control algorithm over drop-tail queues is that sources reduce their transmission rates only after detecting packet loss due to queue overflow. Since a considerable amount of time may elapse between the packet drop at the router and its detection at the source, a large number of numbers of packets may be dropped as the senders continue transmission at a rate the network cannot support.

RED starts to probabilistically drop packets long before the buffer is full, providing early congestion indication to flows which can then gracefully back off before the buffer overflows. RED maintains two buffer

thresholds. When the exponentially averaged buffer occupancy is smaller than the first threshold, no packet is dropped, and when the exponentially averaged buffer occupancy is larger than the second threshold, all packets are dropped. When the exponentially averaged buffer occupancy is between the two thresholds, the packet dropping probability increases linearly with buffer occupancy. It is based on queue length as an estimator of congestion and also requires a wide range of RED parameters to operate correctly under different congestion scenarios. Unfortunately, when a large number of TCP sources are active, the aggregate traffic generated is extremely bursty (3). Bursty traffic often defeats the active queue management techniques used by RED since queue lengths grow and shrink rapidly. While ECN (4) is necessary for eliminating packet loss in the Internet, we show that RED, even when used in conjunction with ECN, is ineffective in preventing packet loss.

FRED extends RED to provide some degree of fair bandwidth allocation. To achieve fairness, FRED maintains state for all flows that have at least one packet in the buffer. The dropping decision is based only on this flow rather than the buffer state in RED. It requires large buffer space to work well. Without sufficient buffer space, it becomes difficult for FRED to detect non responsive flows. (8)(5)

Stabilized RED is used to detect non responsive flows. It keeps a finite log of recent flows it has seen. The non responsive flows will always appear in the log multiple times and can be signaled out for removing. It requires a large buffer space

RED with Per-Flow Queuing takes per-flow queuing and accounting information only for flows which are active. It provides no savings in the amount of state required. If N flows are active, O (N) state must be kept to isolate the flows from each other.(12)

Stochastic Fair Queuing is similar to an SFB queue with only one level of bins. The biggest difference is that, instead of having separate queues, SFB uses the hash function for accounting purposes. Thus, SFB has two fundamental advantages over SFQ. The first is that it can make better use of it buffers. SFB gets some statistical multiplexing of buffer space to individual bins in order to keep the buffer space fully utilized.

III. ACTIVE QUEUE MANAGEMENT ALGORITHMS

A. BLUE

In order to overcome the shortcomings of RED, a fundamentally different queue management algorithm called BLUE implemented. BLUE has been designed with the objective to 1) minimize packet loss rates and queuing delay; 2) avoid global synchronization of sources.

Algorithm:

BLUE uses the packet loss and the link utilization history to manage congestion rather than on the instantaneous or average queue lengths like RED. Also, only a single marking probability is maintained, when the queue is continually dropping packets due to buffer overflow, this probability is incremented, and when the queue is idle or empty, it is decremented. This effectively allows BLUE to learn the correct rate it needs to send back congestion notification. At the same time, the speed of updating of the marking probability depends on a parameter freeze_time. By choosing proper value of freeze_time, we could prevent the p_m from oscillating wildly. Fig. 1 shows the BLUE algorithm.

```
Upon packet loss ( or Qlen > L ) event :
If ( ( now – last_update ) > freeze_time )
                pm = pm + ∂1;
                Last_update = now;
Upon link idle event :
If ( ( now – last_update ) > freeze_time )
                pm = pm - ∂2
                last_update = now;
```

Fig. 1. BLUE algorithm

B. STOCHASTIC FAIR BLUE

Up until recently, the Internet has mainly relied on the cooperative nature of TCP congestion control in order to limit packet loss and fairly share network resources. Increasingly, however, new applications are being deployed which do not use TCP congestion control and are not responsive to the congestion signals given by the network. Such applications are potentially dangerous because they drive up the packet loss rates in the network and can eventually cause congestion collapse. Stochastic Fair BLUE (SFB), a novel technique for protecting TCP flows against non responsive flows. Based on the BLUE algorithm, SFB is highly scalable and enforces fairness using an extremely small amount of state and a small amount of buffer space.

Algorithm:

Fig. 2 shows SFB algorithm. SFB, to detect the non-responsive flows and to protect the other TCP flows from such flows based on accounting mechanisms. Router manages *N x L* bins, and each newly arriving packet is hashed with a different hashing function into one of N bins from L levels. Each such bin maintains a marking/dropping probability p_m as in BLUE. If the number of packets mapped to a bin goes beyond a threshold, p_m is incremented. But if the number of packets drops to 0, p_m is

decreased. The intuition is that non-behaving flows would quickly push the p_m in all the levels to 1. At the same time, for TCP flows, because of low frequency of their packets compared to that of UDP flows, at least one layer will have $p_m < 1$. If a flow has all its $p_m = 1$, it is marked as non-responsive and its rate is limited. Fig. 3 shows an example of SFB of how SFB works.

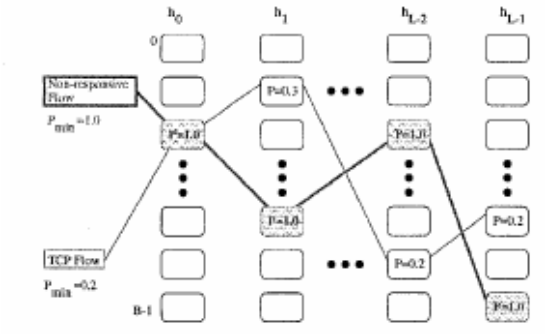

Fig. 2. SFB algorithm

```
Initialization( )
Allocate 23 X 2 array of Bins ( N X L )
( L levels, N bins per each level )
enque( )
Calculate hashes h0, h1….hL -1
Update bins at each level
    For i = 0 to L – 1
    If each level of bin queue
        length > bin_size
                pm = pm + ∂1
                Drop packet
    else if each level of
        bin queue length = 0
        pm = pm – ∂2
        If ( pm = 1)
                Rate limit of non responsive flow else
        Mark probability with Pm
```

Fig. 3. Example of SFB

C. CORE STATELESS FAIR QUEUING

In this section, we know about architecture (Fig. 4) that approximates the service provided by island of Fair Queuing routers, it has a much lower complexity in the core routers. The architecture has two key aspects. First, to avoid maintaining per-flow state at each router, we use a distributed algorithm in which only edge routers maintain per-flow state, while core routers do not maintain per-flow state but instead utilize the per-flow information carried via a label in each packet's header. This label contains an estimate of the flow's rate; it is initialized by the edge router based on per-flow information, and then updated at each router along the path based only on aggregate information at that router.

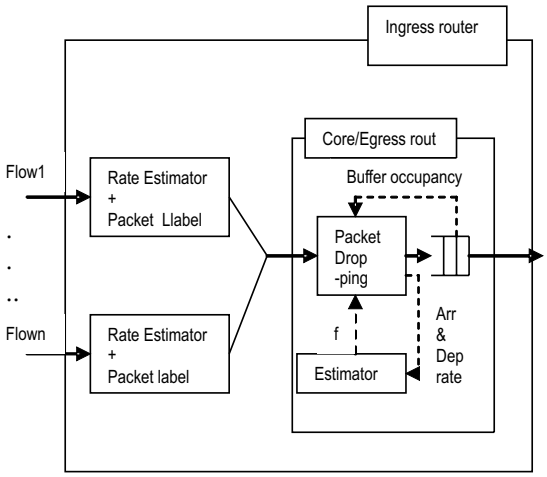

Fig. 4. Architecture of CSFQ

Second, to avoid per-flow buffering and scheduling, as required by Fair Queuing, we used FIFO queuing with probabilistic dropping on input. The probability of dropping a packet as it arrives to the queue is a function of the rate estimate carried in the label and of the fair share rate at that router, which is estimated based on measurements of the aggregate traffic.

The detailed design of this architecture:

1) Computation of Flow arrival Rate: The rates $r_i(t)$ are estimated at the edge routers and then these rates are inserted into the packet labels. At each edge router, use exponential averaging to estimate the rate of a flow. Let t_i^k and l_i^k be the arrival time and length of the kth packet of flow i. The estimated rate of flow I, ri, is updated every time a new packet is received

$$r_i(t) = (1 - e^{-T_i^k/K}) l_i^k / T_i^k + e^{-T_i^k/K} r_{i,old} \quad [1]$$

where,

$$T_i^k = t_i^k - t_i^{k-1}$$

K is Constant

2) Link Fair rate Estimation: When the link is congested, the fair rate *f* is computed such that the rate of the aggregate forwarded rate equals the link capacity.

When the link is uncongested, *f* to be the maximum among the arrival rates of the incoming flows (i.e., the largest label of a packet seen during a certain period).

3) Computation of Packet Forwarding Rate: Upon a packet arrival each node computes its forwarding probability *P* based on the following formula

$$P = \min(1, f/r) \quad [2]$$

where *r* is the current estimated rate of the flow, which is contained in the packet label, and *f* is the fair rate of the

output link. By forwarding the packet with probability *P*, the expected rate of the flow's forwarded traffic is

$$r' = r \times \min(1, f/r) = \min(f, r) \quad [3]$$

4) Label Rewriting: To reflect the eventual change in flow's rate, when a packet is forwarded its label is set to *r'* = min (*f, r*). In this way we ensure the label consistency, i.e., at the next node the label will still represent the estimate rate of the flow's incoming traffic.

IV. SIMULATIONS AND RESULTS

In order to evaluate the performance of BLUE, SFB and CSFQ, a number of simulation experiments were run using ns-2 over a small network shown in Fig. 5 with varying number of input nodes. Using this network, FTP sources were run from one of the leftmost nodes to one of the rightmost nodes. In addition, all sources were enabled with ECN support, were randomly started within the first 1 s of simulation. Packet loss statistics were then measured after 100 s of simulation for 100s.

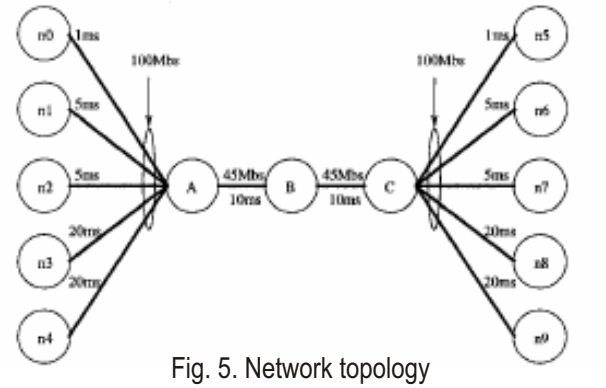

Fig. 5. Network topology

Marking probability of Blue

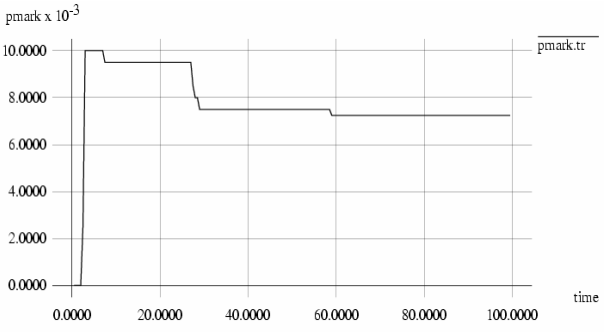

Fig. 6. Marking behavior of BLUE (p_m)

Fig. 6 shows the, the marking behavior of BLUE. The marking probability of RED changes considerably over time and hence cannot remove synchronization among sources. As a result, BLUE marks packets randomly and evenly over time. Consequently, it does a better job in avoiding global synchronization.

Marking Probability of SFB:

In this experiment, 200 TCP sources and one non responsive, constant rate source are run for 100s from randomly selected nodes in Fig. Fig. 8 shows the marking probability of SFB.

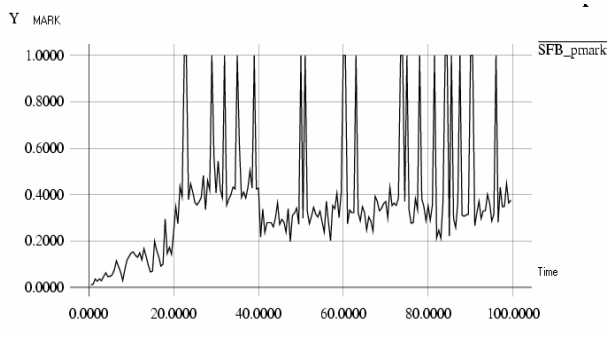

Fig. .8 Marking Probability of SFB

Packet Loss rates of RED and BLUE for 100 sources nodes:

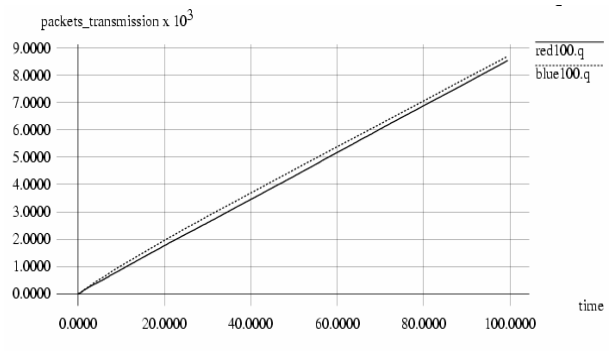

Fig. 7. Packet loss rates of RED and BLUE (100 nodes)

Packet Loss Rates in SFB, CSFQ and FRED:

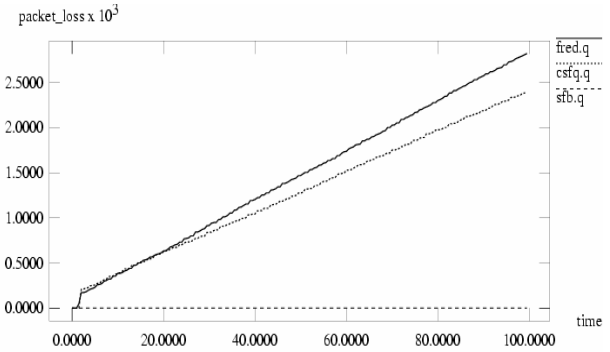

Fig. 9. Packet loss rates of CSFQ, SFB, and FRED

Fig. 9 shows the packet loss rates of FRED, SFB, CSFQ. In this experiment, the loss rates observed over same queue size (128 packets) and 10 Mbps link capacity between A and B nodes in Fig. 5.

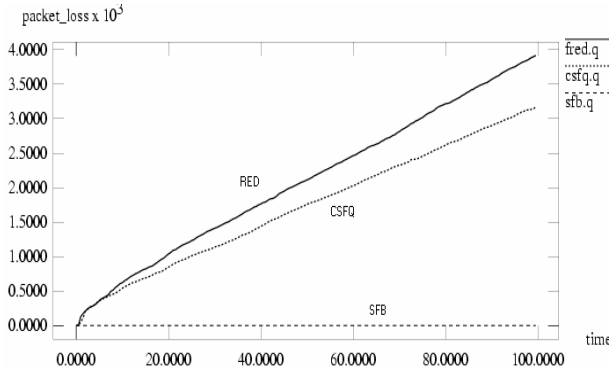

Fig. 10. Packet loss rates of SFB, CSFQ, and FRED

Fig. 10 shows the loss rates observed over different queue size (128 packets for FRED, CSFQ and 50 packets for SFB) and 1 Mbps link capacity between A and B nodes in Fig. 5. Both above experiments prove that the SFB gives better performance than CSFQ and FRED. And also CSFQ gives better performance than FRED.

Bandwidth utilization of SFB, CSFQ and FRED:

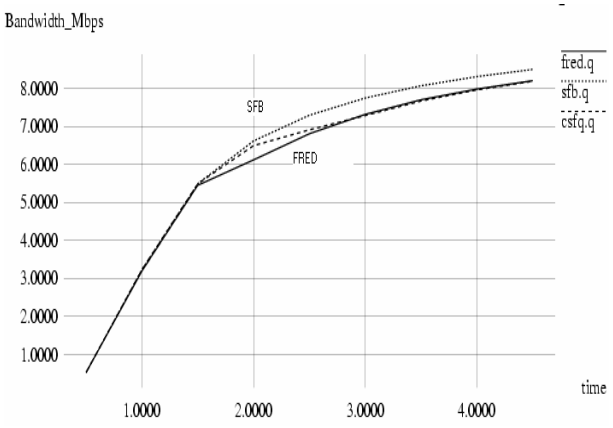

Fig. 11. Bandwidth Utilization of SFB, CSFQ, FRED (1Mbps)

Fig. 11 shows the bandwidth utilization of SFB, CSFQ, and FRED. This figure shows that SFB gives better bandwidth utilization compared with CSFQ and FRED. And also CSFQ makes use of better link capacity compared with FRED. Suppose we increase the link capacity above some extend, either CSFQ or SFB gives better performance rather than the FRED.

Non-Responsive packet loss rates for SFB, CSFQ and FRED:

The Non-Responsive flows are rate-limited to a fixed amount of the link bandwidth. Fig. 12 shows the Non-Responsive packet loss rates for SFB, CSFQ and FRED.

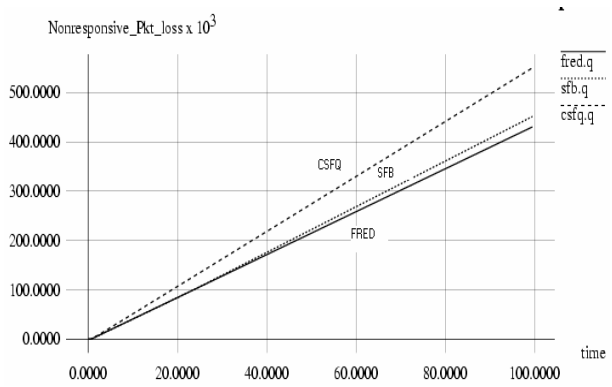

Fig. 12. Non-Responsive Packet loss rates of SFB, CSFQ, and FRED

The above figure shows that CSFQ technique is used to rate limit more number of non-responsive flows compared with SFB. The CSFQ gives better performance rather than the FRED and SFB. But SFB is better than the FRED. In CSFQ, the maintaining of link state information at edge router is simpler job compared with SFB.

V. CONCLUSION AND EXTENSION OF WORK

BLUE uses the packet loss and link utilization history of congested queue, instead of queue lengths to manage congestion. A SFB, a technique using BLUE for scalable and accurately enforcing fairness amongst flows in a large aggregate. Using SFB, non responsive flows can be identified and rate-limited using a very small amount of state. But in case of CSFQ, to use rate estimation at the edge routers and packet labels to carry rate estimates to core routers. Core routers merely perform probabilistic dropping on input based on these labels and an estimate of the fair share rate. Thus, the scheme trades off the overhead in the packet header at every network link. In addition to, it requires both the intermediate router and edge devices adhere to the same labelling and dropping algorithm. A misconfigured or poorly implemented edge device and significantly impact the fair ness of the scheme. SFB, on the other hand, does not rely on coordination between intermediate routers and edge markers and can perform will without placing additional overhead in packet headers. This work will be extended into Differentiated Services Network also.

REFERENCES

[1] W. Feng, D. Kandlur, D. Saha, and K. G. Shin, "The BLUE active queue management", IEEE/ACM transactions on networking, vol 10, No. 4, August 2002.

[2] R. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L.Peterson, K.

Ramakrishnan, S.Shenker, J. Wroclawski, and L. Zhang, "Recommendations on queue management and congestion avoidance in the Internet", RFC 2309, Apr 1998.

[3] W. Feng, D. Kandlur, D. Saha, and K. G. Shin, "Techniques for eliminating packet loss in congested TCP/IP networks", Univ. Michigan, Ann Arbor, MI, Tech. Rep. UM CSE-TR-349-97, Oct. 1997.

[4] S. Floyd, "TCP and explicit congestion notification", Comput. Commun. Rev., vol. 24, no. 5, pp. 10-23, Oct. 1994.

[5] K. Ramakrishnan and S. Floyd, "A proposal to add Explicit Congestion Notification (ECN) to IP", RFC 2481, Jan. 1999.

[6] Ion Stoica, Scott Shenkaer, and Fellow, "A Scalable Architecture to approximate fair bandwidth allocation in high speed networks", IEEE/ACM Transactions on networking, vol. 11, No. 1 February 2003.

[7] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the Internet", IEEE/ACM Transaction on Networking, vol. 7, pp, 458-472, August 99.

[8] S.Floyd and V. Jacobson, "Random early detection for congestion avoidance", IEEE/ACM Transactions Networking, Vol. 1, pp 397-413, July 1993.

[9] V.Jacobson,"Congestion avoidance and control", in Proc. ACM SIGCOMM, Aug 1998, pp. 314-329.

Santhi .V

She has received M.E. degree in Computer Science and Engineering from Anna University in 2004. She published 2 International Conference papers and 2 National Conference papers.