Gateway Algorithm for Fair Bandwidth Sharing

Wei Yi, Rupinder Makkar, Ioannis Lambadaris
Department of System and Computer Engineering
Carleton University
1125 Colonel By Dr., Ottawa, ON K1S 5B6, Canada
{wyi, rup, ioannis}@sce.carleton.ca

Ioannis Marmorkos
Department of Informatics and Telecommunicating
Larisa Institute of Technology
Larisa, 41110, Greece
marmorkos@teilar.

***Abstract*—In this paper we propose a Virtual Round Robin (VRR) gateway algorithm to enforce per-flow fair bandwidth allocation by keeping per-flow information. This mechanism achieves reasonably fair bandwidth allocation and is easily amenable to high-speed implementations. It uses a single FIFO queue with probabilistic drop-on-arrival. In our simulation study, we compare the performance VRR with other two algorithms, Random Early Detection (RED) and Flow Random Early Drop (FRED) [2]. Our simulation results show that VRR outperforms RED and FRED in wide variety of scenarios.**

Keywords-Bandwidth Allocation; Congestion Control; Gateway Algorithm;

I. INTRODUCTION

Mechanisms that can fairly allocate bandwidth in the gateways can provide end users an incentive to be adaptive to network congestion and are critical for Internet congestion control [6][7]. An approach for the gateway would then be to use per-flow queuing and scheduling[4][9]. This approach can ensure fairness, but it does not scale well considering that many flows pass through an Internet gateway. To reduce the complexity and increase efficiency, the IP gateways usually use a single first-in first-out (FIFO) packet queue shared by all flows. They provide feedback to senders by discarding packets under overload. Random Early Detection (RED)[1][5] uses randomization to ensure that all connections encounter the same loss rate. However, this equal drop probability does not lead to equal bandwidth share. Another approach is to use per-flow information, not per-flow queueing and scheduling [2][3]. This approach uses single FIFO queue for packets from all arriving flows, but counts the number of arriving/departing packets of each active flow, and calculates its buffer occupancy. This per-flow information is used to differentiate dropping probability among connections. Flows with different buffer occupancy have different drop probabilities. Flow Random Early Drop (FRED) [2] is the example of this approach. This approach has considerable less complexity than the implementation of per-flow queuing and scheduling, thus provides an intermediate solution between RED and per-flow queuing.

We present a gateway algorithm that provides fair bandwidth allocation to the different flows passing through the gateway. The proposed scheme, Virtual Round Robin (VRR), applies different drop probability to each flow by looking at its state so that each flow is serviced with its fair share and excess packets are dropped. This mechanism achieves reasonably fair bandwidth allocation. As shown in the simulations, VRR can achieve better fairness than FRED. It also has similar complexity as FRED and is easily amenable to high-speed implementations. It uses a single FIFO queue with probabilistic drop-on-arrival; when a packet arrives either the packet is dropped or placed on a FIFO queue. The dropping decisions are simple with $O(1)$ complexity. Contrasting with FRED, the per-flow states of VRR are not the actual buffer occupancy of each flow, but the virtual buffer occupancy of each flow as if all flows were serviced in fair round robin fashion. The gateway algorithm VRR is given in section 2. The algorithms of making drop decision upon packet arrival and of flow states management are introduced. In section 3, the simulation results analysis and performance evaluation are presented. We compared the performance of VRR under a variety of scenarios to two other gateway algorithms: RED and FRED[1][2].

II. VIRTUAL ROUND ROBIN (VRR)

In VRR, the gateway can recognize packets from different flows. All packets accepted will be enqueued to a single FIFO queue. However, the number of packets accepted from each active flow is accounted. For each flow, VRR maintains two per-flow variables, $qlen_i$ and $strike_i$. $qlen_i$ represents the virtual buffer occupancy of flow i and $strike_i$ indicates whether flow i is responsive to congestion. The per-flow state $qlen_i$s of VRR are not the actual buffer occupancy of each flow, but the number of buffered packets as if all flows were serviced in fair round robin fashion. The value of $qlen_i$ is decrease with the fair share rate of flow i, and is increased when the packets from flow i are accepted. Thus, the value of $qlen_i$ represents the size of sub-queue i as if all the flows have their own sub-queue and are serviced in a round-robin fashion. Although all the packets are actually serviced in single FIFO queue, the drop probability of flow i is calculated based on the value of $qlen_i$. Thus, this VRR gateway will have the same drop behavior as the gateway with per-flow queueing and scheduling. Since all the packets accepted will be eventually transmitted, the bandwidth distribution is actually determined by how the packets get dropped. This algorithm can achieve similar performance as gateway algorithms with per-flow

scheduling in bandwidth allocation, but with significantly less complexity.

A. Packet Drop algorithm for VRR

$\min_{th}$: minimum threshold of RED

$\max_{th}$: maximum threshold of RED

$\max p$: maximum dropping probability of RED

w_q: the weight of Exponential Weighted Moving Average

$q\lim$: Buffer size

$\min\_q$=4: the minimum value of $qlen_i$ without packet loss

Global Variables:

$qlen$: total queue size

avg: the average of $qlen$

$\max Flow$: index of the virtual queue that have the most buffered packets

$Nact$: No. of active flow

Per-flow Variables:

$qlen_i$: virtual queue size of flow i

$strike_i$: if the value is 1, flow i can have no more than $\min\_q$ packets buffered

For each arriving packet from flow i :

Calculate the avg;

Compute the drop probability P based on RED algorithm;

//when buffer overflow, mark the flow with most packets buffered as unresponsive

if ($avg \geq \max_{th} \| qlen \geq q\lim$)

$strike_{\max Flow} = 1$;

if ($qlen_i \leq$MAX($\min\_q$, $\frac{\min_{th}}{Nact}$))

Accept the packet from flow i;

//unresponsive flow is not allowed to buffer more than $\frac{\min_{th}}{Nact}$ packets

else if ($strike_i$==1)

drop the packet;

else //operating in random drop mode

drop the packet with the probability P;

B. Algorithm for updating the value of $qlen_i$

In VRR we decrement the value of $qlen_i$s in a round-robin way when the packets are serviced, so that the $qlen_i$ will be the supposed number of buffered packets of flow i if the packets from each flow are dequeued in a fair round-robin way. We do this by keeping an ActiveList, which is a list of indices of non-zero $qlen_i$. The algorithm is:

Whenever a packet from flow i is accepted and enqueued:

$qlen_i$++;

if ($qlen_i$==1) // flow i is a new active flow

append i to the end of ActiveList;

whenever a packet dequeued :

let k be the first element of ActiveList;

$qlen_k$--;

remove the k from ActiveList

if ($qlen_k$>0)

append k to the end of ActiveList;

else discard k;

III. SIMULATIONS

We evaluate the performance of VRR by simulations in ns2 [8]. As a benchmark, we compare the performance of this gateway algorithm to RED and FRED. All data packets in our simulations are 1000 bytes. To set the parameters of RED, we follow the guideline from [10]. The is set to 30 packets and is three times . The is set to 0.002 and the value of is set to 0.1. All the RED gateways have the buffer limit of 200 packets. The VRR and FRED have the same parameters as in the RED setting above, except that the of FRED and of VRR are 4 as recommended in [2].

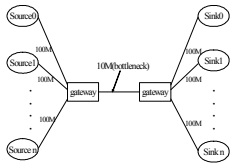

Figure 1 A Single Congested Link

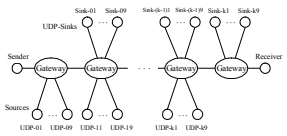

Figure 2 Topology for Multiple Congested Links

To evaluate the performance, we measure the average throughput of each flow in all our simulations. In some cases, we also calculate the Jain's fairness index [11] to measure how effective different gateway algorithms can fairly allocate the bandwidth among flow. The fairness index F is defined as:

$$F = \frac{(\sum_{k=0}^{N-1} b_i)^2}{N\sum_{k=0}^{N-1} b_i^2} \quad (1)$$

Where b_i is the throughput of flow$_i$ and N is the total number of active flows in the gateway.

A. Flows with different sending rates

In this experiment, we test the performance of gateway algorithms when flows with different arrival rate compete at the bottleneck link as in the topology in Figure 1. Under an ideal algorithm, the bandwidth achieved by each flow should be no more than its fair share rate no matter its sending rate. The excess packets from the flow that has higher arrival rate than its fair share will be dropped in the gateway and the flows with a sending rate less than their fair share should experience no packet drop. In this experiment, we have 10 UDP flows,

indexed from 0 to 9. The sending rate of the first flow is the 1Mbps, which is equal to fair share rate, and every subsequent flow sends at a rate 0.2Mbps higher than the previous flow. Thus flow 0 transmits at 1.0 Mbps, flow 1 transmits at 1.2 Mbps, and the last flow transmits at 2.8 Mbps.

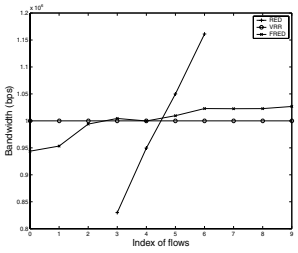

Figure 3. Bandwidth of TCP flows with different sending rate

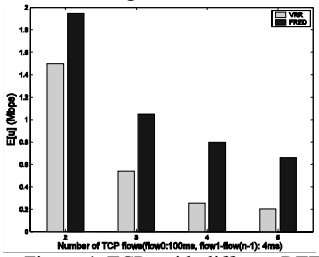

Figure 4. TCPs with different RTT (E(u), mean of u)

Figure 3 summarizes the bandwidth share of each flow gotten when different gateway algorithms are used in the bottleneck link. VRR has the best performance in this scenario. Each flow got exactly its fair share and well-behaved flow0 suffer no packet loss because it sends packets at its fair share rate. On the contrary, bandwidth share of each flow is proportional to its sending rate with RED. FRED is much fairer than RED, all the flows receive roughly their fair share. However, the throughputs of high sending rate flows are still slightly higher than low sending rate flows. The throughput of flow0 is 0.94Mbps in FRED gateway. That means flow0 suffered 6% packet drop rate even though it sends at its fair share.

B. *TCP flows with different RTT*

In this experiment we show the effectiveness of gateway algorithms to protect the TCP flow with large round-trip delay. In this simulation we let n TCP flows traverse the gateway, where flow0 has long RTT(100ms) and the other flows have short RTT(4ms). Because the TCP's bias against the large RTT flows, flow0 normally get less throughput compared to the other flows. We measure u, the difference between long RTT TCP bandwidth and average low RTT TCP bandwidth, where

$$u = \frac{1}{n}\sum_{i=1}^{n-1} B_i - B_0 \qquad (2)$$

n is the number of TCP flows, and B_i is the bandwidth of flow$_i$. In an ideal case, the flow0 with less throughput has low drop probability and its throughput increases. So the value of u should be small. Under VRR and FRED, each simulation is repeated 20 times and the mean value of u, E[u], are computed. As shown in the Figure 4, the mean value of u in VRR is always smaller than in FRED. However, flow0 still gets less throughput than other flows under VRR even though VRR adopt a fair round-robin fashion to decrease $qlen_i$ so that each flow that has non-zero value of $qlen_i$ should obtain its exact fair share. This is because the TCP flow's sending rate is fluctuating while the size of congestion window is changing. For certain period of time, its sending rate is less than its fair share and $qlen_i$ becomes zero. When $qlen_i$ becomes zero, the value of $qlen_i$ cannot be decreased and flow$_i$ will miss its round. From the point of view of gateways, this flow temporarily becomes inactive and cannot claim its share of bandwidth. For all the active flows, the fairness is still maintained.

C. *Mixture of TCP flows and UDP flows*

In the first experiment we show the impact of one non-adaptive UDP flow on a set of adaptive TCP flows. In this simulation, there are 5 flows traverse the bottleneck link. We assume that flow 0 is running non-adaptive UDP while the rest of the flows are adaptive TCP connections. The simulation results are shown in Figure 5.1-5.3. As we can see in Figure 5.1, RED has no control over non-adaptive flow when it competes to adaptive TCP flows. The higher sending rate the CBR flow has, the more bottleneck bandwidth it can obtain. The other TCP flows can only share the leftover bandwidth. With the growth of CBR sending rate to the link rate, TCP flows get very little share.

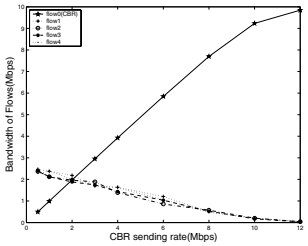

Figure5.1 Flow bandwidths (RED)

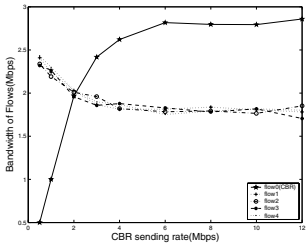

Figure5.2 Flow bandwidths (FRED)

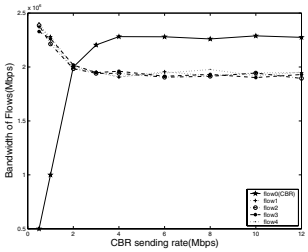

Figure5.3 Flow bandwidths (VRR)

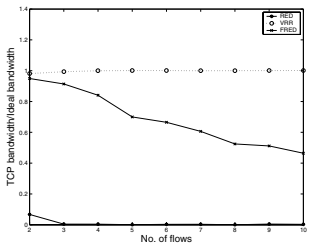

Figure 6 A TCP competing with multiple UDP

Comparing to RED, other active queue management schemes in our experiment can all effectively prevent the non-adaptive flow from taking all the bottleneck bandwidth. In VRR, the CBR flow may get higher bandwidth than TCP flows when its sending rate is higher than 2Mbps as seen in Figure 5.3. However, comparing to Figure 5.2, the performance of VRR is still better than FRED in this scenario. Under VRR flow0 can get no more than 2.3Mbps throughput, which is 15% more than its fair share. Under FRED flow0 will achieve 2.9Mbps throughput, which is 45% more than its fair share.

In the following experiment, we measure how well the algorithms can protect a single TCP flow against multiple non-adaptive UDP flows. We consider experiments involving N flows, $N = 2 \cdots 10$. Out of these we assume N - 1 UDP flows and one TCP flow for each experiment. Each UDP

sends at twice its fair share rate of $\frac{10}{N}$ Mbps. Figure 6 plots the ratio between the average bandwidth of the TCP flow and the fair share bandwidth it should receive. VRR has the best performance across the entire range; TCP flow can achieve its ideal fair share. In RED, TCP flow is completely shut out by the non-adaptive flows and obtains no bottleneck bandwidth. FRED can also protect the TCP flow effectively from being shut out. However, it is harder for TCP flow to get its fair share under FRED as the number of competing UDP flows increases. In case of 9 competing UDP flows, the TCP flow only achieves 46.4% of its fair share.

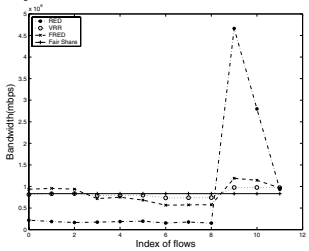

Figure 7. Mix of 9 TCP flows and 3 UDP flows

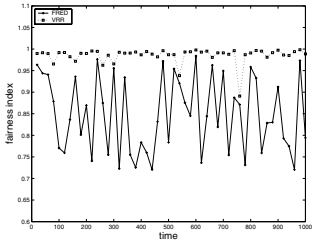

Figure 8. Fairness index of each 20ms period for mix traffic

D. *Mixture of heterogeneous flows*

This experiment has a mix of TCP and CBR UDP flows. There are 9 TCP flows and 3 CBR flows. The TCP flows have different round-trip times; the first 3 TCP have round-trip times close to 20ms, the next three have RTTs around 40ms, and the last three have RTTs around 60ms. The CBR flows, with flow numbers 9, 10, 11, have sending rate of 5 Mbps, 3 Mbps and 1Mbps respectively. Figure 7 shows the bandwidth of each of the 12 flows with RED, with FRED and with VRR. With RED, the high-bandwidth CBR flows get almost all the bandwidth, leaving little for the TCP flows. All the flow experience the same loss rate, so the flows with higher sending rate will achieve higher bandwidth. As seen in Figure 7, under RED flow9 gets 4.7Mbps bandwidth which is close to its 5Mbps sending rate. Both VRR and FRED are able to restrict the bandwidth received by the CBR flows to near the fair share. However, with VRR all the flows have a more evenly distributed bandwidth share.

In a subsequent simulation, we compare the performance of VRR when the traffic rates are randomly changing. We use the same traffic combination as above. We assume intervals of 20 seconds each. During each interval a flow has 60% probability to transmit. We run FRED and VRR as gateway algorithms and compute the fairness index of each 20s time interval. As shown in figure 8, the fairness index of VRR is consistently higher that that of FRED. Also the value of fairness index under VRR is much smoother than the one under FRED. The average value of fairness index of VRR is 0.986, and confidence interval is [0.981, 0.991] The average value of fairness index of FRED is 0.848, and confidence interval is [0.822, 0.874]

E. *Multiple congested links*

We now analyze how the throughput of a well-behaved flow is affected under different gateway algorithms when the flow traverses more than one congested link. The simulations are performed based on the topology shown in Figure 2. Except the flow from the sender to receiver, all flows are 2Mbps UDP flows. The capacity of each link in the system is 10 Mbps, and each link between gateways has 9 UDP flows at 2 Mbps sending rate. This will cause all links between gateways being congested.

In the first simulation, we let a UDP flow sending at its fair share rate of 1Mbps from Sender to Receiver. In an ideal case, this flow should suffer no packet loss and have all its traffic forwarded because it sends at its fair share. Figure 9 shows the fraction of UDP flow that is transferred versus the number of congested links. VRR has the best performance in this case. With the increase of the number of congested links, the throughput of well-behaved UDP flow is always close to its fair share bandwidth. FRED also performs significantly better than RED. The VRR outperforms the FRED in the entire range in Figure 9. When the number of congested links is 10, the well-behaved UDP flow under FRED achieves only 82% of its fair share, while under VRR it obtain its full fair share. There is an 18% difference in bandwidth sharing between FRED and VRR in this case.

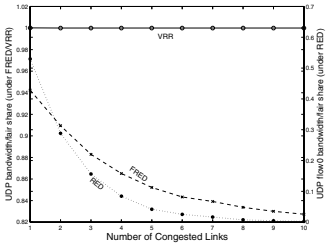

Figure 9 A well-behaved UDP flow Traversing multiple congested links

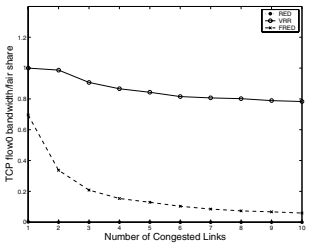

Figure 10 TCP flow traversing multiple congested links

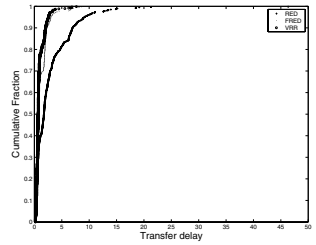

Figure 11 Cumulative fraction of web transfer delay

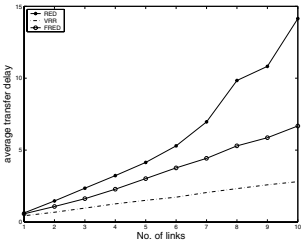

Figure 12 average web transfer over multiple congested links

In another simulation, we show the behavior of TCP flow traversing multiple congested links. We use the same topology as in Figure 2, and send a FTP flow from sender to receiver instead of a UDP flow. In an ideal gateway, this test flow should experience very little packet loss at every congested link when its arrival rate is lower than its 1Mbps fair share. And this low drop rate will allow the TCP congestion window of this connection grow, so this flow can achieve its 1Mbps fair share throughput. The actual throughput of test TCP flow under different algorithms is shown in Figure 10, when TCP flow traverses multiple congested links, its throughput decreases in all gateway algorithms with the number of

congested links because its packets are more likely to be dropped. As an extreme case, TCP flow gets completely shut down when the congested gateway uses RED. Under FRED, the throughput the TCP flow decreases significantly with the number of congested links increase. VRR outperforms FRED over all range. Under VRR, flow0 achieve 78.3% of its fair share when flow0 traverse 10 congested links, it achieves 78.3% of its fair share under VRR and only 5.9% under FRED.

*F. **Short-lived web transfers***

In this simulation we assess the performance of VRR with short-lived web traffic. Short-lived web flows spend significant amount of time in the slow start phase, so they send at low rate most of the time. A fair gateway algorithm should be able to reduce the loss rate of web traffic and allow the short flow to finish sooner. The background traffic in the simulation is the mix of long lived FTP flows and unresponsive UDP flows. There are 9 FTP flows with different RTT and 3 CBR flows with 5Mbps, 3Mbps, and 1Mbps sending rate respectively. In addition, we consider 600 web-objects transfer with average of 20 packet per web-object to transfer. We use the built-in web-traffic model in NS. The competing flows starts randomly within the initial 5s of simulation while the web-traffic start after 50s. We record the object transfer delays for each web transfer. Figure 11 is the cumulative distribution of transfer delay of web-traffic under different gateway algorithms. Both FRED and VRR can reduce the loss rate of web traffic compared to RED. So the transfer delay is smaller in these two algorithms. The results are summarized in Table below:

Algorithm	Transfer delay	Std. dev	Loss rate
RED	2.1123	2.8882	5.36%
FRED	0.6344	0.3997	0.84%
VRR	0.5976	0.3011	0.52%

Because the loss rate in VRR is less than in FRED, the transfer delay under VRR is less than under FRED.

As the experiment above shown, both FRED and VRR can protect low bandwidth web-transfer flow. So this two algorithm can reduce the transfer delay of short web traffic. We now study the performance of these algorithms when the web connections traverse multiple congested link. We use the topology of Figure 2. Each congested link has 10Mbps capacity and 10 FTP background connections. The short web-traffic connections start after 50s simulation time from the sender to receive. Each simulation is repeated 5 times and the mean loss rate and transfer delay are the average of all simulations. We record the average object transfer delay for web transfers in figure 12. As shown in the figure 12, as the number of congested links increase, the mean transfer delays are increasing. This is because the packets from web-traffic are more likely to be dropped when they go through multiple congested links. Under VRR the average transfer delays of web traffic are smaller than FRED in all range of figure 12. This shows that VRR has better protection of low bandwidth flows than FRED.

IV. CONCLUSION

We have suggested VRR, a gateway algorithm that enforce fairness among active flows by differentiate their drop probabilities according to the flow. This approach is suitable for high-speed implementations because it use a single FIFO queue for all incoming packets with probabilistic drop-on-arrival. It can allocate bandwidth fairly and has similar complexity as FRED.

The performance of VRR is compared with RED and FRED in our simulation studies. In all cases, VRR and FRED achieve significant performance improvement over RED because of extra flow states used. With the same complexity as FRED, VRR outperform FRED

REFERENCES

[1] S. Floyd, V. Jacobson, Random Early Detection for Congestion Avoidance, IEEE/ACM Transactions on Networking 1(4), August 1993, 397-413.

[2] D. Lin, R. Morris, Dynamics of Random Early Detection, Proceeding of ACM Sigcomm, ACM, New York, 1997, pp. 127-137.

[3] F. Anjum, L. Tassiulas, "Fair Bandwidth Sharing among Adaptive and Non-Adaptive Flows in the Internet." Proc. IEEE INFOCOM, Mar. 1999, 1412-1420.

[4] E. L. Hahne, Round robin scheduling for fair flow control in data communications networks, IEEE JSAC, vol. 9, no. 7, Sept. 1991, pp. 1024-1039

[5] B. Braden et al., Recommendations on Queue Management and Congestion Avoidance in the Internet, RFC 2309, IETF, April 1998

[6] G. Hasegawa, M. Murata, Survey on Fairness Issues in TCP Congestion Control Mechanisms, IEICE Trans. Commun., June 2001, Vol.E84-B, No.6.

[7] A. Mank, K. Ramakrishn, Gateway Congestion Control Survey, RFC1254, IETF, August 1991

[8] Network Simulator. http://www.isi.edu/nsnam/ns/

[9] M. Shreedhar, G. Varghese, "Efficient fair queuing using deficit round robin", IEEE/ACM Trans. Networking, June 1996

[10] Sally Floyd, "RED: Discussions of Setting Parameters", www.icir.org/floyd/REDparameters.txt

[11] R. Jain, "The Art of Computer Systems Performance Analysis", New York: Wiley-Interscience, 1991