Fair Resource Allocation in Hybrid Network Gateways with Per-Flow Queueing

Roshni Srinivasan, Ravichander Vaidyanathan, John S. Baras
Center for Satellite and Hybrid Communication Networks
University of Maryland
College Park, MD 20742 *

Abstract

In this paper, we present an efficient resource allocation scheme for scheduling and buffer management in a bottleck hybrid internet gateway. We use Fair Queueing in conjunction with Probabilistic Fair Drop, a new buffer management policy, to allocate bandwidth and buffer space in the gateway to ensure that all TCP flows threading the gateway achieve high end-to-end throughput and fair service. We propose the use of buffer dimensioning to alleviate the inherent bias of the TCP algorithm towards connections with large Round Trip Time and validate our scheme through simulations.

1 Introduction

In recent years, the feasibility of using existing internet protocols in the TCP/IP suite in hybrid satellite-terrestrial networks has become an active research area. Transmission Control Protocol (TCP) is an adaptive window based protocol used for flow and congestion control over the internet. The sliding window used by the TCP algorithm at the source dynamically varies in response to acknowledgements, timeouts or packet losses. When multiple TCP connections share a link with high bandwidth-delay product, it has been observed [1] that synchronization of multiple TCP connections results in low link utilization. The nature of the algorithm causes severe degradation of TCP throughput when a connection loses multiple packets. Recent simulation studies [2] have shown that per-flow scheduling and buffer management policies, in spite of their higher computational complexity and control overhead, perform better than their global counterparts, which consider aggregated flows.

*This work was supported by the Center for Satellite and Hybrid Communication Networks, under NASA cooperative agreement NCC3-528

TCP's inherent unfairness to connections with long Round Trip Time (RTT) is especially evident when a connection that includes a long latency satellite hop shares a bottleneck link with other connections that have relatively smaller RTT's as illustrated in fig. [1].

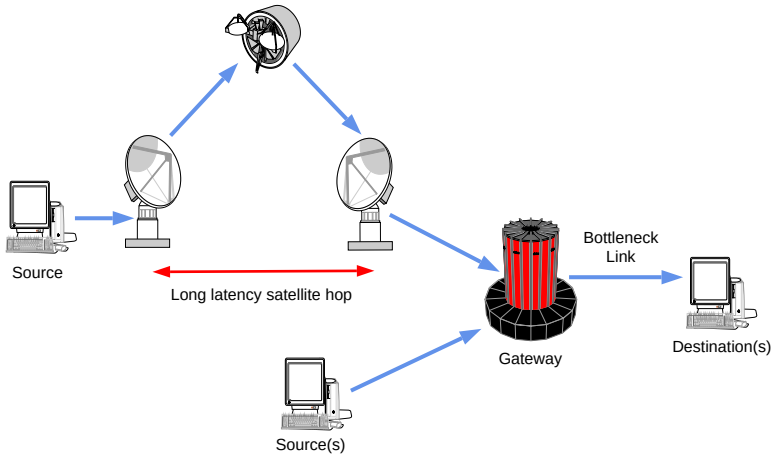

Figure 1: Motivation

The traditional internet, with its characteristic "best-effort" service now carries traffic of widely varying throughput, delay, jitter or loss requirements. In order to provide service differentiation between these different traffic classes, network bandwidth needs to be managed efficiently. The mechanism that facilitates this management is the scheduling algorithm used in intermediate routers or gateways. The choice of the scheduling discipline at these switching nodes determines the per-connection end-to-end performance guarantees that the network can provide. Important factors that determine this choice are the efficiency of the algorithm in enforcing the Quality of Service (QoS) guarantees, the computational complexity and the fairness and flexibility of the algorithm in handling excess traffic.

While fair scheduling ensures efficient use of network bandwidth, the role of buffer management becomes evident at the time of network congestion. Packets must be dropped in accordance to a policy that provides isolation (protection from misbehaving sources), fairness (proportional use of available buffer) and efficiency (decision to discard a packet). In this paper, we propose the use of *Probabilistic Fair Drop* (PFD), a buffer management strategy, which in combination with an efficient fair scheduling algorithm will improve TCP throughput significantly. We compare the performance of our scheme with that of other buffer management schemes such as Random Early Detection (RED) [7], Longest Queue Drop (LQD) and Random Drop (RND) [2].

2 Fair Scheduling : An Introduction

Fair scheduling algorithms that provide end-to-end delay guarantees on a per-flow basis try to emulate the ideal behavior of the Generalized Processor Sharing Algorithm (GPS) which is based on a fluid flow model of traffic. The GPS policy is ideally fair in handling excess traffic. If $W_i(t_1, t_2)$ is the amount of service flow i recieves in the interval (t_1, t_2), GPS ensures that for any two flows, i, j back-logged during the interval (t_1, t_2), the following relation holds

$$\frac{W_i(t_1, t_2)}{\phi_i} = \frac{W_j(t_1, t_2)}{\phi_j} \tag{1}$$

where ϕ_i and ϕ_j are the weights assigned to the flows i, j respectively.

Weighted Fair Queuing(WFQ) [3] and Packet-by-Packet generalized Processor Sharing (PGPS) [4] are approximations to GPS scheduling that do not make GPS's infinitesimal packet size assumption. The end-to-end delay bound resulting from PGPS can be computed based on the weight assigned to the flow. The complexity of the insertion and deletion from sorted priority queue is $O(logN)$, making it infeasible for a very large number of flows. Worst-case Fair Wieghted Fair Queueing (WF^2Q) uses both the start and finish times of packets in the reference GPS system to achieve a more accurate emulation of GPS. Self-Clocked Fair Queuing (SCFQ) [5] is a way to speed up PGPS's biggest limitation - the round-number computation. The end-to-end delay bound of SCFQ, however, is much larger than PGPS. In our study, we use Start-time Fair Queuing (SFQ) [6] as the scheduling discipline. SFQ uses the efficient computational aspects of SCFQ but does not have the large worst case delay and short term unfairness suffered by SCFQ. In SFQ, packets are serviced in the increasing order of start tags, the assignment of which is *O(1)* in computational complexity. SFQ is computationally efficient, has a bounded fairness measure and achieves low average as well as maximum delay for low throughput connections.

3 Related Approaches to Buffer Management

The "random drop" notion of RED gateways was motivated by the systematic discrimination against some connections by a "Drop-Tail" gateway in a TCP/IP network with strongly periodic traffic. The decision to accept an incoming packet in RED is based on whether the low pass filtered average queue size exceeds or conforms to predetermined thresholds. As long as the average queue size is between min_{th} and max_{th}, an arriving packet is dropped with a probability that increases linearly with average queue size. Flows are penalized roughly in proportion to their buffer occupancy. While RED tries to ensure per-flow fairness without per-flow state having to be maintained in the router, it turns out to be unfair to low rate

TCP flows. The randomness of the drop decision combined with the dropping of the arriving packet could cause the congestion window of a flow to be halved irrespective of whether it exceeds its fair share of the buffer. Flow RED (FRED) [8] was proposed to alleviate some of RED's limitations. FRED monitors buffer occupancies on a per-flow basis with min_{th} and max_{th} being defined for each flow. This rigid allocation of thresholds, however, is not effective in managing excess bandwidth when the network is not congested.

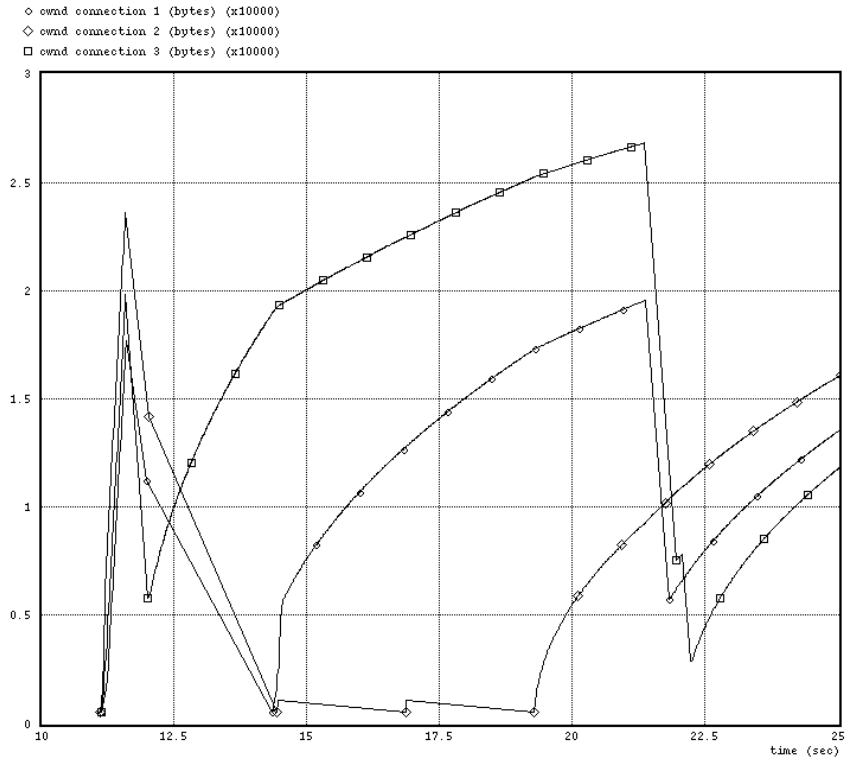

Figure 2: Multiple drops in RND : connection 1 and connection 3 suffer drops with high temporal locality and drop their windows simultaneously. connection 2 suffers multiple drops initially and experiences severely degraded performance.

The notion of having higher packet drop rates for connections with longer queues is reflected in the LQD and RND policies of [2]. Each flow is allocated a "soft threshold" b_i with the total buffer capacity given by $B = \sum_{i=1}^{n} b_i$. The instantaneous queue size q_i is monitored for each flow. These schemes, unlike RED and its variants, do not drop the incoming packet. When the total buffer occupancy, $\sum_{i=1}^{n} q_i$ is B is exceeded, a packet is chosen to be discarded from a connection whose current occupancy exceeds its allocation. In LQD, a packet is dropped from a connection i for which $(q_i - b_i)$ is the largest. In RND, the connection i is chosen at random from amongst connections for which $(q_i > b_i)$. Once the connection is chosen, the packet is dropped from the front of the queue in order to trigger TCP's FRR mechanism faster [9]. While being innovative, these schemes necessarily have to drop packets during congestion. They do not try to predict the onset of congestion by monitoring the queue size and use the instantaneous queue size to determine the drop queue.

By waiting until congestion occurs, it is highly likely that multiple packet drops occur with high temporal locality, leading to one of two undesirable behaviors.

1. The same source is penalized multiple times, leading to severe degradation in the throughput of this source (LQD).
2. Multiple sources are penalized, causing flow synchronization and hence degradation in the link utilization(RND).

In summary, and to motivate our approach, we make the following observations on existing buffer management schemes.

1. Predicting the onset of congestion helps avoid multiple drops with high temporal locality.
2. Per flow schemes such as FRED however, are rigid in their threshold allocations, and do not allow a flow to expand into the global buffer space even when buffer space is available.
3. Schemes with a global threshold such as RED are unfair to low rate TCP flows.

4 The Probabilistic Fair Drop (PFD) Algorithm

In the presence of per flow queuing, we dynamically share the total available buffer B equally among all currently active connections n. A soft threshold b_i is allocated for each flow $b_i = \frac{B}{n}$. Later sections discuss buffer dimensioning to alleviate TCP's unfairness to long RTT connections.

We attempt to predict the onset of congestion by monitoring the total instantaneous buffer occupancy $q\_size = (\sum_{i=1}^{n} q_i)$ against a single global threshold, $thresh$. As long as the buffer occupancy does not exceed $thresh$, no packets are discarded and a flow may expand to fill all the available buffer space. Once $thresh$ is exceeded, a drop decision is evaluated with a fixed probability p. This decision is independent of the choice of the flow to be penalized.

If the decision is made to discard a packet, a flow is chosen as follows. We define a normalized instantaneous flow size, $n_i = q_i/b_i$. We now choose the flow with the highest normalized instantaneous flow size and *push out* the packet at the head of the chosen flow. The drop probability, p and threshold, $thresh$ prove to be effective measures to counter the effects of multiple packet losses for a single connection as well as flow synchronization.

Specified Parameters :

p : fixed drop probability
$thresh$: fixed threshold (% of global buffer space) $(0 < thresh < 1)$
B : global buffer space

Variables :

q_size : global instantaneous buffer occupancy
q_i : instantaneous occupancy of queue i
n_i : normalized instantaneous occupancy of queue i
b_i : dynamic soft threshold of queue i
$drop\_q$: queue to discard from i

Algorithm :

For each packet arrival
    increment q_size
    classify packet into flow i
    increment q_i
    compute $n_i = \frac{q_i}{b_i}$

if $q\_size > B$
    $drop\_from\_longest\_normalized\_q$
else if $q\_size > thresh * B$
    with prob p
    $drop\_from\_longest\_normalized\_q$
else continue

$drop\_from\_longest\_normalized\_q$
    choose $drop\_q = \overset{argmax}{i} \ (n_i)$
    push-out the packet at the head of $drop\_q$
    decrement q_size

Figure 3: The Probabilistic Fair Drop Algorithm

Probabilistic discard with a conservative threshold and a low value of p ensure that a bursty connection will suffer packet losses that are further apart on the time axis, and not be penalized repeatedly. This also allows more time for the connection to respond to congestion. Unlike RND, where packets necessarily had to be discarded from a set of flows when the buffer overflowed, with PFD, packet discards from different sources occur with low temporal locality, thus preventing flow synchronization.

Packet drops in PFD are not always from non-conformant sources, thus providing an early warning mechanism for conformant sources with large queue buildups as well. Since PFD penalizes the queue which utilizes its largest normalized buffer share, the drop proba-

bility for a queue is proportional to its normalized buffer occupancy.

5 Simulation Methodology and Results

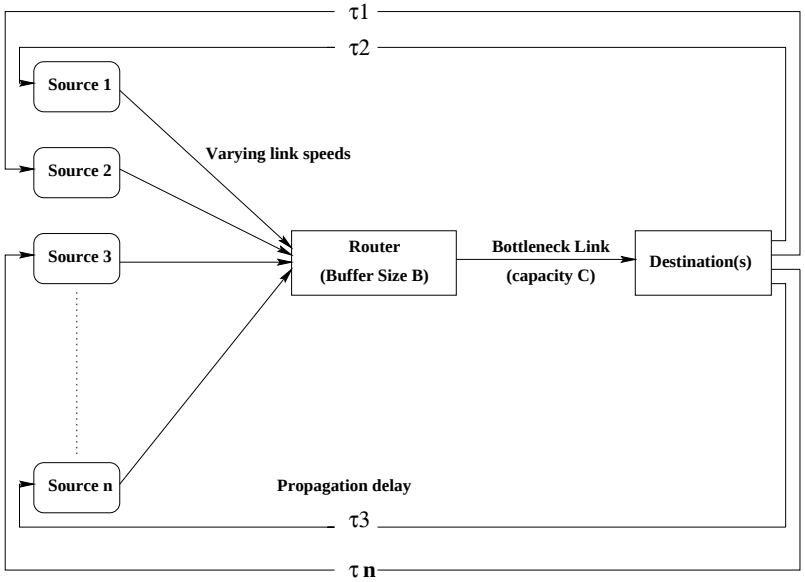

Figure 4: Simulation Model

Simulations were carried out using the **OPNET** Network simulator [12]. We adopt the n source TCP configuration for the network model. In this configuration the TCP sources share a common bottleneck link of capacity C and are subjected to varying propagation delays. The TCP sources implement TCP Reno with the Fast Retransmit and Fast Recovery algorithms [13]. In this section, we use data sources with very large file sizes to compare the steady state performance of PFD versus that of other buffer management schemes. Studies on short transient connections are documented in a later section.

The router at the bottleneck link implements packetized versions of several buffer management strategies including Tail Drop, Drop from Front, RED, LQD, RND and PFD. Scheduling strategies such as FCFS, Round Robin and FQ are also configurable at this router. Queuing effects are limited to the IP queues in the gateway by matching the IP forwarding rate at the router with the desired bottleneck link rate C.

In our simulations, we studied the performance of the PFD buffer management scheme with a FQ scheduler when multiple TCP connections with different rates and propagation delays share a bottleneck link with high bandwidth delay product. The router at this bottleneck link is configured with various buffer management and scheduling policy combinations

used for the study. We compare the performance of FQ-PFD with FCFS-RED, FQ-RED, FQ-LQD and FQ-RND.

5.1 Performance Measures

We evaluate the performance of the various schemes using the TCP "goodput". We define the *goodput* as the average rate of data delivered to the application by TCP. The *total scalar goodput* is obtained by averaging the goodput across all connections. To evaluate the steady state performance lare data sources are used for long periods of time until the average rate stabilizes.

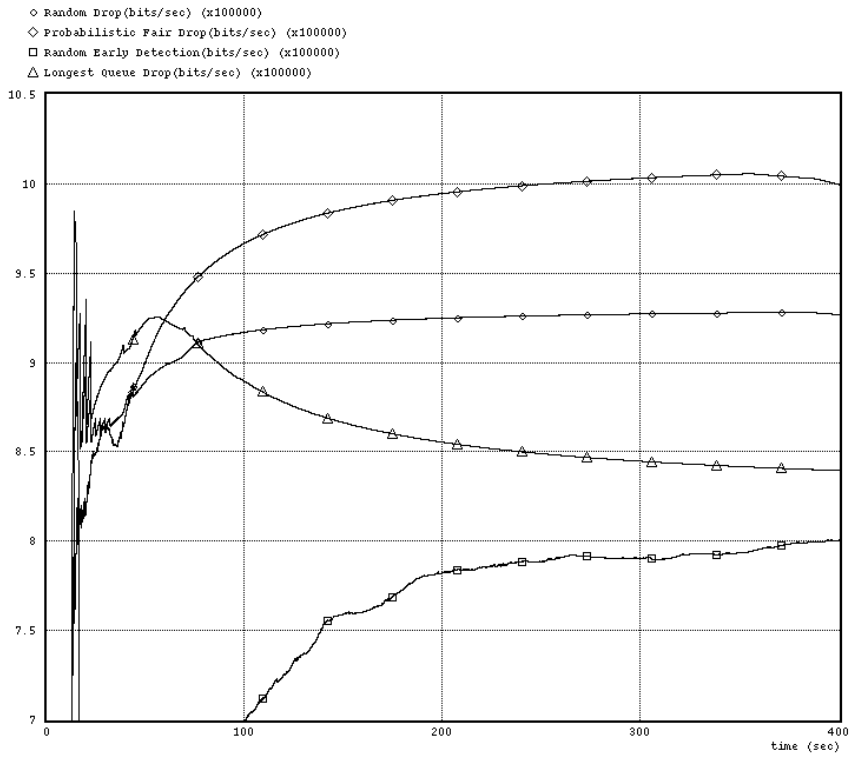

Figure 5: Evolution of TCP goodput with time for 10 low RTT TCP-reno connections sharing a bottleneck link of 2 Mbps

The ideal combination of scheduling discipline and buffer management policy would ensure that each flow threading the bottleneck router receives the same amount of service over any interval of time. A good scheme would penalize a flow that exceeds its fair share and ensure that excess bandwidth is shared fairly among backlogged flows when the network is not congested. We use the *Fairness Coefficient* defined in [10] as a measure of how fairly the scheme distributes bandwidth among competing flows. The Fairness Coefficient, F is

defined as

$$F = \frac{(\sum_{k=1}^{N} b_i)^2}{n \sum_{k=1}^{N} b_i^2} \tag{2}$$

where n is the number of flows and b_i is the bandwidth obtained by $flow\ i$. From this definition, we see an ideally fair scheme would have a fairness coefficient of 1, while a completely unfair scheme would result in a coefficient of $1/n$. We study the fairness of our scheme over different buffer sizes and compare performance with the other schemes.

5.2 Performance results

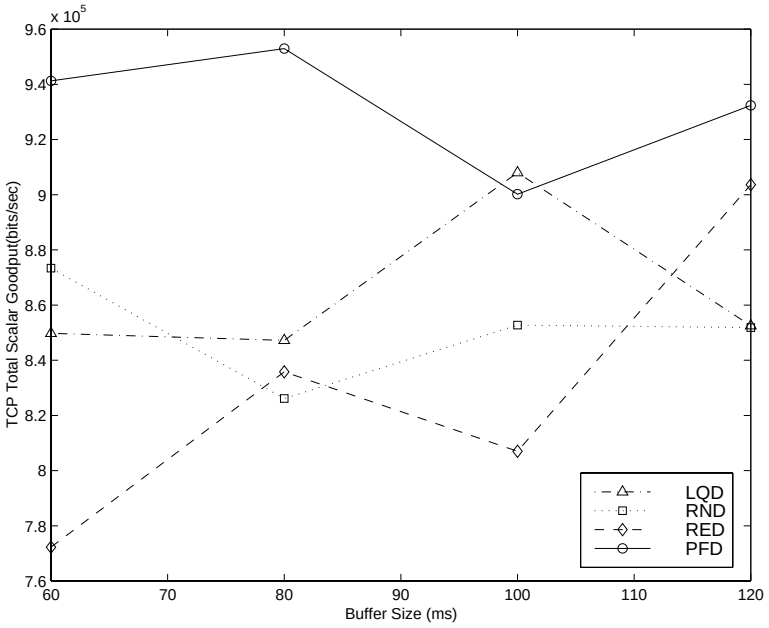

Figure 6: TCP goodput (after 500 sec) versus buffer size at the bottleneck router for 10 low RTT TCP-reno connections sharing a bottleneck link of 2 Mbps

We compare the performance of RED, LQD and RND with PFD with an SFQ scheduler. From fig. [5] we see that in steady state PFD outperforms RED,LQD and RND. The *early warning* system of PFD combined with the pushout drop policy allows PFD to ramp up to a higher average rate.

Fig. [6] shows that PFD in general outperforms other schemes even in the case of short RTT connections sharing a bottleneck link. For this simulation setup it was observed that all the schemes have a fairness coefficient of close to 1. This may be attributed to the similar RTTs and behavior of the connections sharing the bottleneck link.

We now turn our attention to connections with widely varying RTTs. We consider the

goodput of 10 TCP-reno connection with RTTs varying from 20 ms to 200 ms (of the order of a satellite RTT) sharing the same bottleneck link. Note that our version of TCP-reno also implements the TCP window scale option [14] and hence the goodput of the large RTT connections are not limited by the TCP congestion window growth.

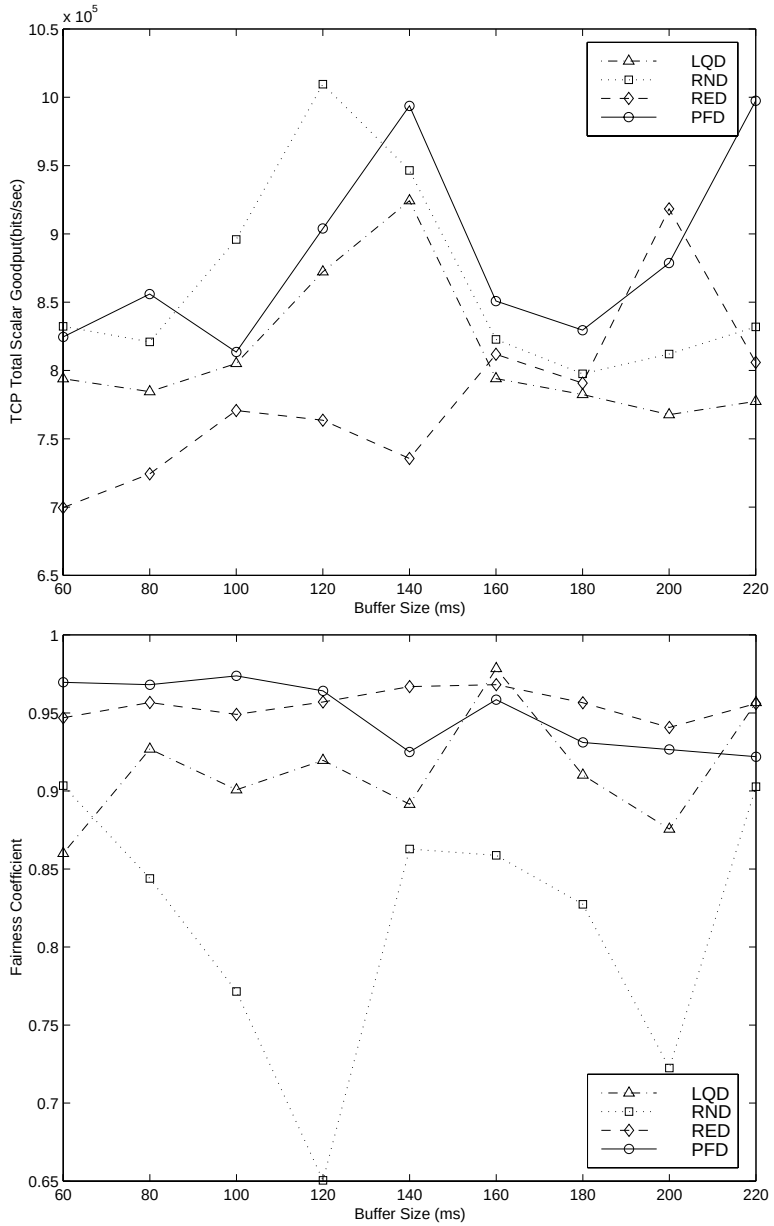

Figure 7: TCP goodput and Fairness Coefficient (after 500 sec) versus buffer size at the bottleneck router for 10 TCP-reno connections with RTTs varying between 20 and 200 ms

We plot the performance over a widely varying range of buffer sizes and attempt thereby to capture the goodput and the fairness metrics over this range. The range of buffer sizes is chosen to capture regions in which the queuing delay is insignificant in comparison to the longest RTT as well as regions in which the queuing delay is significant and is comparable

to the RTT of the longest RTT connection.

From fig.7 we see that for lower buffer sizes RND performs better than PFD in terms of goodput. A comparision with the fairness coefficient curve indicates, however, that the improved goodput is at the expense of fairness to long RTT connections. This may be attributed to the random choice that RND makes from the set of non-coformant connections when choosing a flow to penalize. Not surprisingly, LQD is observed to be consistently fairer than RND. PFD's higher fairness coefficient can be explained by observing that while both LQD and PFD are similar in flavor when picking a flow to penalize, the early drop nature of PFD allows longer RTT flows more time to react.

As expected, RED experiences the worst performance in terms of goodput. The high degree of fairness of RED in this case may be explained by observing that like PFD, RED also employs an early detection mechanism and the random choice of flow appears to even out over longer periods of time. The fairness of PFD becomes more evident over shorter time scales. Also, for asymmetric channels studied in [2] and later in this paper, RED performs very poorly in terms of fairness due to its drop tail nature. We note that PFD offers the best performance from a combined fairness and goodput perspective. In PFD, fairness is achieved without a corresponding decrease in goodput. We further verify our results by studying a mix of 10 TCP-reno connections, 5 of which experience short RTTs and 5 of which experience satellite delay RTTs. The results are presented in and lend themselves to similar observations.

5.3 Buffer Dimensioning to improve the performance of long RTT flows

An analysis of TCP throughput as a function of loss probability [11] has shown that TCP throughput varies inversely as the Round Trip Time (RTT). The dynamics of the algorithm also causes the rate of window growth to be inversely proportional to RTT, both in the Slow Start and Congestion Avoidance phases of the algorithm. This inherent bias towards long RTT connections is further worsened by global buffer management strategies, that do not allow long RTT connections to build up the larger windows they need in order to maintain the same throughput as shorter RTT connections. First Come First Served (FCFS) scheduling, which offers service to a connection in proportion to its buffer occupancy, is known to be more unfair to long RTT connections than Fair Queuing (FQ). In this paper, we address both these issues. We examine the throughput of long RTT connections when FQ is used in conjunction with proportional buffer allocation and PFD. The "soft" per-flow buffer thresholds are allocated in proportion to RTT, with b_i now being allocated as

$$b_i = \{RTT_i/(\sum_{i=1}^{n} RTT_i)\}B$$

This proportional allocation of soft thresholds allows longer RTT connections a better chance to build up their larger windows. PFD is used for packet discard decisions. The assignment of equal weights to all connections ensures that the scheduler still treats all connections with equal priority, giving each a fair share of the link bandwidth.

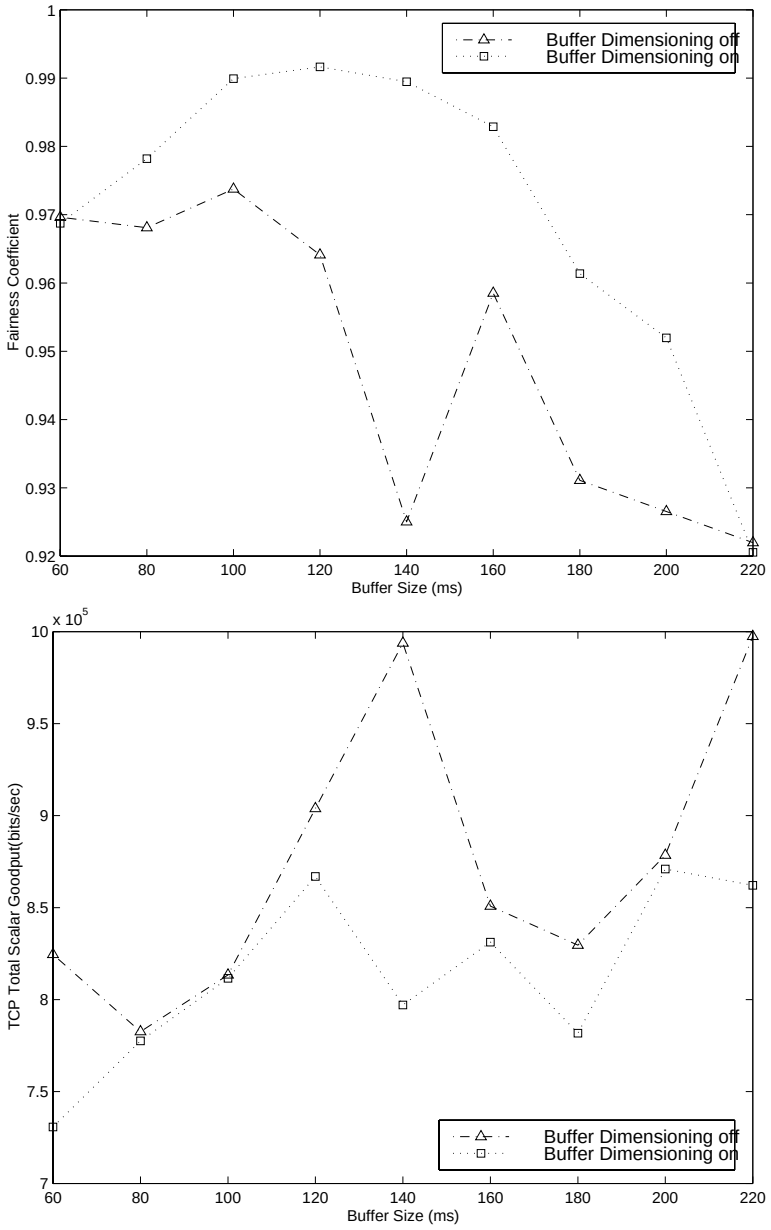

Figure 8: TCP goodput and Fairness Coefficient (after 500 sec) versus buffer size at the bottleneck link with and without buffer dimensioning

Fig. [8] illustrates the effects of buffer dimensioning as outlined earlier and raises the issue of a trade-off between fairness coefficient and TCP goodput. While the proposed scheme results in increased fairness towards connections with long RTT (and therefore an increased

fairness coefficient), this comes at the cost of a decrease in TCP goodput. The overall decrease in TCP goodput resulting from buffer dimensioning is offset by the increased goodput for the higher RTT connections. This trend is favourable from the point of view of individual flows threading a satellite gateway, where a user is charged for the duration of time that the uplink bandwidth is utilized.

An interesting observation in this context is the decline of the fairness coefficent with increasing buffer size. For the same traffic conditions, an increase in buffer size implies fewer packet drops. As the buffer size approaches infinity, there are vitually no discards.The TCP throughput for a connection is then determined by its RTT. The dynamics of the TCP algorithm then come into play and a bias towards long RTT connections is observed inspite of Fair Queueing.

6 Conclusions

In this paper, we study the performance of an efficient resource allocation scheme in a bottleneck hybrid internet gateway with per-flow queueing. We combine a fair scheduling algorithm with Probabilistic Fair Drop, a new buffer management scheme and use simulations to evaluate the end-to-end performance parameters of TCP goodput and Fairness Coefficient for the scheme. Our simulations indicate that this allocation scheme performs well with respect to both these parameters in comparison to other schemes that use Fair Queueing in conjunction with other buffer management policies such as Random Early Detection (RED), Longest Queue Drop (LQD) and Random Drop (RND). We study the use of buffer dimensioning as a method to counter the unfairness of TCP towards connections with large RTT. We conclude from our simulations that the degradation in overall throughput is compensated by increased fairness to longer RTT connections and therefore better throughput for such connections. While the fixed parameters for the PFD scheme worked extremely well for our simualtion set-up, parameter estimation for a wide range of traffic conditions would be required to make the scheme more viable in a dynamically changing environment. We intend to study methods of estimation of drop probability and threshold as a function of buffer size and traffic characteristics in the course of future work.

References

[1] T.V. Lakshman, U. Madhow, "The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss", *IEEE/ACM Transactions on Networking*, pp. 336-350, June 1997.

[2] B. Suter, T.V. Lakshman, D. Stiliadis, A. K. Choudhury, "Design considerations for supporting TCP with per-flow queuing", *IEEE INFOCOM '98*.

[3] A.Demers, S. Keshav, S. Shenker, "Analysis and Simulation of a Fair Queuing Algorithm", *Proc ACM SIGCOMM*, pp. 1-12, 1989.

[4] A.K. Parekh, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks", *Ph.D. thesis, Dept. of EECS, MIT*, LIDS-TH-2089, February 1992

[5] S.J. Golestani, "A Self-clocked Fair Queuing scheme for broadband applications",*Proc. IEEE INFOCOM*, pp. 636-646, June 1994.

[6] P.Goyal, H. Vin and H. Cheng, "Start-time Fair Queuing : A Scheduling Algorithm for Integrated Services Packet Switching Networks", *Proc. ACM SIGCOMM*, pp 157-169, August 1996.

[7] S. Floyd and V.Jacobson, "Random Early Detection Gateways for Congestion Avoidance", *IEEE/ACM Transactions on Networking*, Vol1, No.3, pp 397-413, August '93.

[8] D. Lin and R. Morris, "Dynamics of random early detection", *Proceedings of SIGCOMM*, pp 127-137, September 1997.

[9] T.V. Lakshman, A. Nierhardt and T.J. Ott, "The Drop from front strategy in TCP over ATM and its Interworking with other Control Features", *Proc. IEEE INFOCOMM*, pp.1242-1250, 1996.

[10] D. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks", *Computer Networks and ISDN Systems*, Vol. 17, No.5, pp1-14, 1989.

[11] J. Padhye, V. Firoiu, D. Towsley, J. Kurose,"Modeling TCP Throughput : A Simple Model and its Empirical Validation", *Proc. ACM SIGCOMM '98*

[12] OPNET Modeler/Radio release 4.0A, *URL http://www.mil3.com*

[13] W.Stevens,"TCP Slow Start, Congestion Avoidance, Fast Retransmit and Recovery Algorithms",*RFC 2001, Jan '97*

[14] V.Jacobson,"TCP Extensions for High Performance", *RFC 1323, May '92*