Fair Queuing with Round Robin: A New Packet Scheduling Algorithm for Routers

Arunabha Sen and Ibraz Mohammed
Dept. of Computer Science
Arizona State University
USA
{asen, ibraz}@asu.edu

Ravikanth Samprathi
R & D Division
CISCO Systems
USA
rsamprat@cisco.com

Subir Bandyopadhyay
School of Computer Science
University of Windsor
CANADA
subir@cs.uwindsor.ca

Abstract

In the last few years several queuing policies have been proposed to ensure fairness between competing requests at a service point. Fair Queuing (FQ) algorithm due to Demers, Keshav and Shenkar is a queuing technique that attains near perfect *fairness, where* perfect *fairness is considered to be the one attained by a ¤uid ¤ow model. In data network,* head of the line processor sharing *(PS) is considered to be the most* fair *algorithm. It has been shown that the difference in throughput at any time, in any queue, for any arrival pattern between the FQ and the PS discipline will never exceed MAX, where MAX is the maximum packet size. This difference in throughput is taken as a metric for* fairness measure *of a queuing algorithm. The drawback of the FQ algorithm is its high packet processing overhead ($O(log\ N)$), where N is the number of active ¤ows. To alleviate this problem of high computational complexity, Shreedhar and Varghese proposed a fair queuing algorithm based on the idea of* de£cit round robin *(DRR). Although DRR reduces the packet processing overhead to $O(1)$, its fairness measure is considerably worse (3MAX) than that of FQ (MAX). In this paper, we present a new round robin based fair queuing algorithm (FQRR) whose packet processing overhead is $O(1)$ and fairness measure is 2MAX.*

1 Introduction

The packet scheduler in a router plays an important role in determining how effectively the QoS service requirements are met by the packets belonging to a ¤ow. Two simple packet scheduling algorithms are First Come First Serve (FCFS) and Round Robin (RR) [5]. These two algorithms however, cannot guarantee fair allocation of network resources in cases of contention between competing resource requests by the ¤ows. To address the issue of *fairness*, several policies, such as Fair Queuing (FQ) and Weighted Fair Queuing (WFQ) [3], De£cit Round Robin (DRR) [6], Worst-case-Fair Weighted Fair Queuing (WF^2Q) [1], have been proposed in the last few years.

The FQ and WFQ algorithms proposed in [3] received considerable attention in the research community due to their *near perfect* fairness (established in [4]) and *delay bound*. Since the algorithm has to maintain a priority queue to determine the packet to be scheduled next, its complexity is $O(log\ N)$ where N is the number of active ¤ows through the router. To alleviate the problem of high packet processing cost of FQ, Shreedhar and Varghese in [6] proposed the low overhead De£cit Round Robin algorithm. Although DRR reduces the packet processing cost to $O(1)$, its fairness measure (difference in throughput between the idealized PS scheme and DRR) is considerably worse (3MAX) than that of FQ (MAX), where MAX is the maximum packet size.

Our goal in undertaking this research was to attempt to develop a fair queuing algorithm that would have the excellent fairness measure of the FQ algorithm (MAX) and low packet processing overhead of DRR ($O(1)$). Although the Fair Queuing with Round Robin (FQRR) algorithm being proposed in this paper does not quite realize this objective, we believe it makes an important step in that direction. We show that FQRR has the low overhead of DRR ($O(1)$), and its fairness measure is 2MAX, which is not as good as FQ (MAX) but is better than that of DRR (3MAX). Considering the fact that DRR is an important fair queuing algorithm that can be used in a high speed router (as evidenced by the use of DRR in CISCO's Gigabit Switch Router 12016), we believe that FQRR may be used in applications where DRR is used. We show that not only FQRR has a better worst-case fairness measure (2MAX as compared to 3MAX), it consistently outperforms DRR in a simulation environment. Our simulation results show that the average fairness measure for FQRR varies between 0.81MAX to 1.64MAX, whereas for DRR it varies between 2.76MAX to 2.89MAX.

2 Prior Work

The idea of using round-robin scheduling for fair allocation of bandwidth was originally proposed by Nagle [5]. In his scheme the gateways maintained separate queues for packets from individual ¤ows. The queues are serviced in a round-robin fashion. The goal was to prevent a ¤ow from arbitrarily increasing its share of the outgoing link bandwidth or the delay of other ¤ows. Although at a £rst glance, the scheme appears to have provided a satisfactory solution to the fairness issue but on closer examination it reveals a serious ¤aw. This is due to the fact that in a large heterogeneous network packet sizes are not required to be identical. As round-robin servicing of the queues can be done in constant time [6], the complexity of this algorithm is $O(1)$.

Demers, Keshav and Shenkar in [3] proposed a clever technique to overcome the problem associated with Nagle's algorithm. They introduced a notion of *virtual time* such that corresponding to any *real time* t there is a virtual time $R(t)$ associated with it. According to their scheme, the rate of change of the virtual time is inversely related to the number of *active queues* at that time. Speci£cally,

$$\frac{d}{dt}R(t) = \frac{1}{max\{1, N_{act}(t)\}} \tag{1}$$

where $N_{act}(t)$ is the number of active queues at time t.

The *virtual start time* and *virtual £nish time* of each packet are computed on its arrival at the queue. Suppose that the (real) arrival time of the i-th packet of α-th queue is τ_i^α, $R(\tau_i^\alpha)$ is the virtual time corresponding to real time τ_i^α and P_i^α is the (normalized) length of the packet. Then the virtual start and £nish times of the i-th packet of α-th queue (S_i^α and F_i^α respectively) are computed by the following recurrence relations, with $S_0^\alpha = F_0^\alpha = 0$.

$$S_i^\alpha = max\{F_{i-1}^\alpha, R(\tau_i^\alpha)\} \tag{2}$$
$$F_i^\alpha = S_i^\alpha + P_i^\alpha \tag{3}$$

There are two different versions of the FQ algorithm: (i) FQ with earliest start time (FQS) and (ii) FQ with earliest £nish time (FQF) [4]. In FQS, when the output link becomes free after servicing a packet, it selects the packet with the earliest virtual start time from the set of packets waiting for service. In the case of FQF, it selects the packet with the earliest £nish time. Since the packet with the earliest start time or the earliest £nish time has to be selected from the set of packets at the heads of various queues, the computational complexity of this algorithm is $O(log\ N)$, where N is the number of queues. Greenberg and Madras in [4] have established the following result for both the versions of the FQ algorithm.

Algorithm	Fairness Measure	Complexity
RR [NAG87]	∞	$O(1)$
FQ [DEM89]	MAX	$O(log\ N)$
DRR[SHR96]	3MAX	$O(1)$
FQRR	2MAX	$O(1)$

Table 1.

Theorem 1 *For all possible realizations of the arrival process, for every time t and for every queue α: $|B(\alpha, PS, 0, t) - B(\alpha, FQ, 0, t)| \leq MAX$, where $B(\alpha, PS, 0, t)$ and $B(\alpha, FQ, 0, t)$ are the throughputs between times 0 and t for the service disciplines PS and FQ respectively.*

Although the fairness measure of the FQ algorithm is excellent, due to its high packet processing overhead ($O(log\ N)$), this scheme may be not be very suitable for high speed networks. To reduce the overhead of the FQ algorithm, Shreedhar and Varghese in [6] proposed a variation of Nagle's round-robin algorithm. Instead of processing exactly one packet from each queue as in Nagle's algorithm, they associate a *quanta* of service for each queue during each round-robin cycle. In addition, they also introduce a notion of *de£cit*. Suppose that the size of the quanta is Q bits, and during a round-robin cycle only S bits ($S < Q$) from a queue α are serviced, even though packets were waiting for service in this queue. In this case, during the next cycle, the size of the quanta for this queue is incremented by an amount $Q - S$, known as the *de£cit*. The complexity of this algorithm ($O(1)$), is signi£cantly lower than that of the FQ algorithm ($O(log\ N)$). However, this bene£t of reduced complexity is off-set by the reduction in fairness measure. The fairness measure of this algorithm is 3MAX, whereas for the FQ algorithm it is MAX.

In spite of the reduced fairness measure of the DRR algorithm (3MAX compared to MAX of FQ), because of its low packet processing overhead, the algorithm is used in one of the fastest routers available in the market today (CISCO's Gigabit Switch Router 12016). The FQRR algorithm being proposed here, has the same low overhead of DRR ($O(1)$), but it is more fair than DRR. The computational complexity and the fairness measures of a few scheduling algorithms are given in table 1.

3 Fair Queuing with Round Robin

Fair Queuing with Round Robin (FQRR) is a round-robin algorithm which utilizes some of the concepts developed in connection with the Fair Queuing algorithm by Demers *et. al* in [3]. The notion of *virtual time* developed in [3] is used in FQRR. As soon as the i-th packet in queue α arrives, the FQRR algorithm computes its *virtual start time,* S_i^α and *virtual £nish time,* F_i^α. A *virtual time quantum,* Q is allocated to each queue for each cycle of the round-robin process. The size of this quantum is set equal to (real) time needed to process the largest sized packet. If a server speed is normalized and it is assumed that it services at 1 bit per second and the largest sized packet has *MAX* bits, Q is taken as *MAX*. The queues are served in a round robin fashion. During the k-th round of the round-robin cycle, all the packets in queue $\alpha, (1 \leq \alpha \leq N)$, whose virtual £nishing time is at most $k * Q = k * MAX$ are served. To discriminate active queues from the inactive ones, the algorithm maintains a data structure called the *ActiveList* like the DRR algorithm in [6]. All active queues are linked to the *ActiveList* and removed from the *ActiveList* when the queue becomes inactive. This prevents the router from having to check the head of up to N packet queues before £nding an active queue that can actually be served. We present the FQRR algorithm next and explain the operation of the algorithm with the help of an example.

Initialization:

begin
    for $(i := 0;\ i < N;\ i = i + 1)$
        begin
            S_0^i = 0;
            F_0^i = 0;
        end
        K = 0;
        Q = MAX;
end

Enqueue Module: (On arrival of i-th packet in queue α)

begin
    Enqueue the i-th packet in queue α;
    $R(\tau_i^\alpha)$ = virtual time corresponding to real time
    of arrival τ_i^α;
    S_i^α = maximum $[F_{i-1}^\alpha, R(\tau_i^\alpha)]$;
    F_i^α = $S_i^\alpha + P_i^\alpha$; (*P_i^α is the length of the packet*)
    If queue α does not exist in the *ActiveList* then
        Insert queue α into *ActiveList*;
end

Dequeue Module:

begin
    while (TRUE)
    begin
        $j[m] = 1,\ \ 1 \leq m \leq N$; ($j[m]$ is the packet at the head of queue m)
        $K = K + 1$;
        $Q = K * MAX$;
        for every queue i in the *ActiveList*
        begin
            while $(F_{j[i]}^i \leq Q)$
            begin
                dequeue $j[i]$-th packet at the head of the queue i;
                $j[i] = j[i] + 1$;
            end
            If queue i is empty then
                Remove queue i from the *ActiveList*;
        end
    end
end

	Queue α		*Queue* β		*Queue* γ
	Pkt 1	Pkt 2	Pkt 1	Pkt 2	Pkt 1
Real arrival time τ_i	0	2	1	2	3
Transmission time P_i	3	1	1	4	3
Virtual start time S_i	0	3	1	2	2
Virtual £nish time F_i	3	4	2	6	5

Table 2.

Example: We use the same example as in [4]. Suppose that the arrival times, packet sizes and the queues are as given in table II. The virtual start times and virtual £nish times of all the packets in all the queues are computed and are shown in the table. Suppose that the maximum packet size is 4 bits. Assuming a server that serves at 1 bit per unit of time, $Q = MAX = 4$. According to the FQRR algorithm, during cycle 1, all the packets in all the queues whose virtual £nish times are at most 4 will be serviced. Therefore, in this example the following packets are serviced during the £rst cycle:

- Queue α: The £rst and the second packets of queue α are served because $F_1^\alpha = 3$ and $F_2^\alpha = 4$.
- Queue β: The £rst packet of queue β is served because $F_1^\beta = 2$.
- Queue γ: No packet is served as there is no packet with virtual £nish time at most 4.

The second cycle begins at the completion of the £rst cycle and during this cycle all the packets in all the queues whose virtual £nish times are at most 2*4 = 8 are served. Therefore, in this example the following packets are serviced during the second cycle:

- Queue α: No packet is served as the queue is empty.

- Queue β: The second packet of queue β is served because $F_2^\beta = 6$.
- Queue γ: The £rst packet of queue γ is served because $F_1^\gamma = 5$.

The progress of packets through the server for PS, FQS, FQF, DRR and FQRR is shown in £gure 1. In the £gure, t represents real time, $R(t)$ represents virtual time.

4 Analytical Results

The analytical results presented in this paper follow closely the techniques used in [4]. Accordingly, we follow the de£nitions and notations used in [4].

De£nitions and Notations:

N: Maximum number of queues (Queues are numbered from 1 through N)
$N_{act}(t)$: The number of active queues at time t
τ_i^α: The arrival time of the i-th job in queue α
P_i^α: The length of the i-th packet in queue α
MAX: Maximum packet size
$R(t)$: The *virtual time* corresponding to *real time t*
$R^{-1}(t)$: The *real time* corresponding to *virtual time t*
S_i^α : Virtual start time of the i-th job in queue α
F_i^α : Virtual £nish time of the i-th job in queue α
PS: Head of the line *Processor Sharing* queuing discipline
FQRR: *Fair Queuing with Round Robin* queuing discipline
$B(\alpha, D, s, t)$: The number of bits serviced from queue α during the real time interval $[s, t)$ using queuing discipline D
Q: Virtual time quantum allocated for each round of the round-robin process
σ_i^α : Start time (real) of the i-th job in queue α in FQRR discipline
ϕ_i^α : Finish time (real) of the i-th job in queue α in FQRR discipline
$n_\alpha^{FQRR}(t)$: The number of packets serviced in queue α by time t in FQRR

Lemma 2 *The PS system is busy at real time t if and only if the FQRR system is busy at time t. More speci£cally, for every time t,*

$$\sum_{\omega=1}^{N} B(\omega, PS, 0, t) = \sum_{\omega=1}^{N} B(\omega, FQRR, 0, t)$$

Proof: The proof of this lemma has not been included here for the sake of brevity. The proof is given in [7].

Lemma 3 *For every α and t,*
$B(\alpha, PS, 0, t) - B(\alpha, FQRR, 0, t) \geq -2MAX.$

Proof: The proof of this lemma has not been included here for the sake of brevity. The proof is given in [7].

Lemma 4 *For every α and t,*
$B(\alpha, PS, 0, t) - B(\alpha, FQRR, 0, t) \leq 2MAX$

Proof: The proof of this lemma is similar to that of lemma 3, but it has been not included here for the sake of brevity. The proof is shown in [7].

Theorem 5 *For all possible realizations of the arrival process, for every time t and for every queue α:* $|B(\alpha, PS, 0, t) - B(\alpha, FQRR, 0, t)| \leq 2MAX$

Proof: Follows from lemmas 3 and 4.

5 Computational Complexity

In this section we present the packet processing overhead associated with the FQRR algorithm and compare it with the DRR algorithm.

(i) When a packet arrives at the input link of the router, FQRR checks to see if this ¤ow is already in the $ActiveList$. This requires one basic operation (a comparison).

(ii) If the queue is not part of the $ActiveList$ then FQRR adds this queue to the $ActiveList$. One elementary operation is needed for this purpose. But this overhead is not associated with every single packet, but only with the packet that turns an inactive queue to an active one. Therefore, the overhead of this operation may be amortized over many packets and consequently, the overhead associated with an individual packet may be negligible in that situation. However, in a worst-case scenario, only one packet may have this overhead associated with it. Accordingly, the overhead associated with each packet can vary from 0 to 1 basic operation.

(iii) In the FQRR algorithm the *virtual start time* and the *virtual £nish time* of each packet are computed on its arrival at the queue using equations 2 and 3 given in section II.

The (real) arrival time of the i-th packet of α-th queue, τ_i^α is known. If both F_{i-1}^α and $R(\tau_i^\alpha)$ are known, then S_i^α can be computed with only one comparison operation. Since F_{i-1}^α is computed at the time of arrival of the $(i-1)$-th packet in queue α, it is already known at the time S_i^α is being computed. Next we £nd the number of operations needed to compute $R(\tau_i^\alpha)$.

In the router only two types of *events* take place - *packet arrival* and *packet departure*. As a consequence of these events, the *number of active queues* may or may not change. One variable can be used to keep track of the number of active queues and updated whenever necessary. If the *virtual time* is plotted against the *real time* the gradient of this

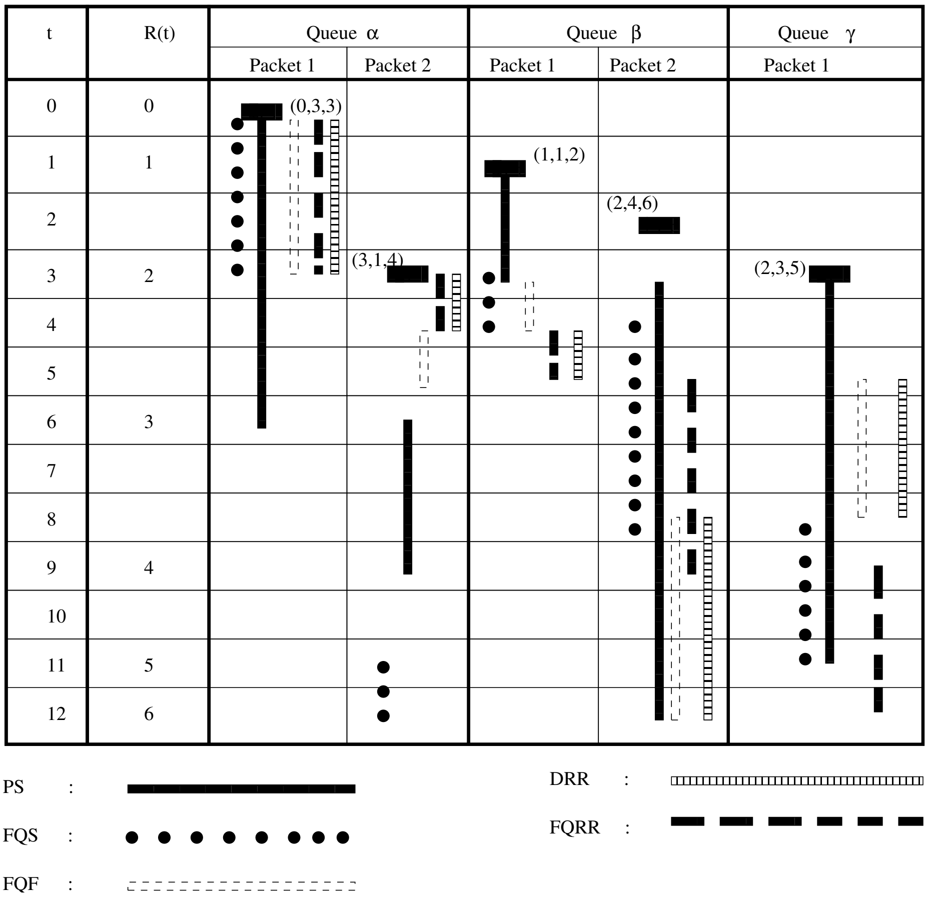

Figure 1. Packet processing in PS, FQS, FQF, DRR and FQRR. The boxes mark packet arrival times. The (S, P, F) triplets next to the boxes indicate the virtual start times, packet lengths and virtual £nish times of the packets.

line will change only at times when the number of active queues change. The virtual time $R(\tau_i^\alpha)$ corresponding to real time τ_i^α can be computed as follows: Suppose that σ is the latest time prior to τ_i^α when the number of active queues changed and $N_{act(\sigma)}$ is the number of active queues between the instances of time σ and τ_i^α. In this case $R(\tau_i^\alpha) = R(\sigma) + (\tau_i^\alpha - \sigma)/max\{1, N_{act(\sigma)}\}$.

Therefore, the computation of $R(\tau_i^\alpha)$ will require 4 elementary operations (an addition, a subtraction, a division and a comparison). Once $R(\tau_i^\alpha)$ is computed, S_i^α can be computed with one additional comparison operation. Finally, F_i^α can be computed with one more addition operation. Thus S_i^α can be computed with £ve operations and F_i^α can be computed with one additional operation.

(iv) When a particular round of service is initiated, the round number (K) is incremented and the virtual time quantum (Q) is updated. This constitutes two elementary operations (an addition and a multiplication). But this computation is done once for each round and not for each packet. Therefore, the overhead is distributed among the packets serviced from all the queues active during the current round. Accordingly, the overhead associated with each packet can vary from 0 to 2 basic operations.

(v) When a particular ¤ow α has to be serviced, the algorithm checks to see if the virtual £nish time F_i^α of the packet at the head of the queue α is less than or equal to quantum Q for this round. This constitutes one elementary operation (a comparison). If the condition is satis£ed then the packet is removed and is sent to the output link for transmission.

(vi) The algorithm then checks to see if queue i has become empty. This requires one basic operation (comparison). If the queue has become empty then the queue is removed from the $ActiveList$. This may require one additional operation (an assignment). Although two elementary operations are needed if the queue is empty, they are not associated with every single packet, but associated with all the packets that are serviced during a round. Accordingly, the overhead associated with each packet can vary from 0 to 2 basic operations.

Thus the total overhead associated with processing of a packet in the FQRR algorithm (measured in terms of the number of basic operations) varies from 8 to 13.

A similar analysis shows that the total overhead associated with processing of a packet in the DRR algorithm (measured in terms of the number of basic operations) varies from 6 to 12.

The number of basic operations for the FQRR algorithm is 8 to 13 and for the DRR algorithm it is 6 to 12. In the worst case scenario FQRR requires one more operation than DRR; and in the best case scenario, two more operations are required. In view of the fact that the fairness measure of FQRR is better (2MAX) compared to DRR (3MAX), this additional overhead seems to be adequately justi£ed.

6 Simulation Results

The objective in undertaking the simulation experiments was to check if the worst case bounds of the fairness measure for the FQRR (2MAX) and the DRR (3MAX) are attained in practice. Although our analytical results showed that the maximum difference in throughput of PS and FQRR service disciplines is 2MAX, and in the case of DRR, it is 3MAX, it is conceivable that in practice, the difference in throughput between the PS and the FQRR is A.MAX and between the PS and the DRR is B.MAX, where B is less than A. In other words, we wanted to check if in practice the performance of these two disciplines comes anywhere near their worst case behavior.

Our simulation is for a single router. The maximum number of queues considered is 20. The arrival process is Poisson. The packet sizes are drawn from a uniform distribution between 0 and MAX, where MAX is the maximum packet size. We consider two different scenarios, in the £rst scenario there are no ill-behaved sources and in the second scenario there are two ill-behaved sources. The simulation is conducted for £fty different values for the arrival rate λ. The simulation is carried for 200 units of time, and the result of the difference in throughput between PS and FQRR (DRR) is noted at time $t = 10, 20, \ldots, 200$ for 20 different queues. $MaxDiff$ is the maximum difference in throughput between either PS and FQRR or PS and DRR taken over all twenty queues and over all 20 instances of time $t = 10, 20, \ldots, 200$. The experiments are conducted for 10 different values for MAX and as such we have 10 such tables, which are not presented here for brevity. The results are available in [7].

We have computed the mean and the standard deviation of the maximum difference in throughput between PS and FQRR, and PS and DRR. For the results presented in the table in [7], the mean of the maximum difference in throughput between PS and FQRR is 1.63678MAX and between PS and DRR is 2.855349MAX, when there is no ill-behaving source. In the case where ill-behaving sources are present, the values are 1.64863MAX and 2.847862MAX respectivly. In absence of ill-behaving sources, for different packet sizes, the results varied from 0.806968MAX to 1.63678MAX for PS-FQRR and 2.790318MAX to 2.89866MAX for PS-DRR . In presence of ill-behaving sources the differences varied from 0.806825MAX to 1.648630MAX and 2.766610MAX to 2.893551MAX respectively.

As predicted by our analytical results the difference in throughput between PS and FQRR never exceeded 2MAX in our 160,000 experiments. Similarly, as predicted in [6] the difference in throughput between PS and DRR never exceeded 3MAX. The fact that the simulation results are completely compatible with the analytical results enhances the con£dence of the analysis. From these simulation results, we also learn that the maximum difference in throughput between PS and FQRR is considerably lower (1.63678MAX) than that of its worst case bound (2MAX), whereas the difference in case of PS and DRR can actually reach close (2.89866MAX) to its worst case bound of 3MAX. From these perspectives, it appears that the FQRR scheme is a much *fairer* service discipline than DRR.

References

[1] J. C. R. Bennet and H. Zhang,"WF^2Q: Worst-case fair weighted queueing", *Proc. of IEEE INFOCOM'96*, pp. 120-128, San Francisco, March 1996.

[2] J. C. R. Bennet and H. Zhang,"Why WFQ is not good enough for Integrated Services Networks", *Proc. of NOSS-DAV, Shonan Village International Conference Center, Zushi, Japan*, April. 1996.

[3] A. Demers, S. Keshav and S. Shenkar,"Analysis and simulation of a fair queuing algorithm", *Journal of Internetworking Research and Experience*, pp. 3-26, October 1990. Also in *Proc. of SIGCOMM'89*, pp. 3-12.

[4] A. G. Greenberg and N. Madras, "How fair is fair queuing", *Journal of the Association of Computing Machinery*, vol. 39, no. 3, pp. 568-598, July 1992.

[5] J. B. Nagle, "On packet switches with in£nite storage", *IEEE Transactions on Communications*, COM 35, no. 4, pp. 435-438, 1987.

[6] M. Shreedhar and G. Varghese, "Ef£cient fair queuing using de£cit round robin", *IEEE/ACM Transactions on Networking*, vol. 4, no. 3, pp. 375-385, June 1996. Also in *Proc. of SIGCOMM'95*, pp. 231-243, Boston, MA, September 1995.

[7] A. Sen, R. Samprathi, I. Mohammed and S. Bandopadhyay "Fair Queuing with Round Robin: A New Packet Scheduling Algorithm for Routers", Technical Report, Dept. of Computer Science & Engg., ASU, 2001.