FRR: a Proportional and Worst-Case Fair Round Robin Scheduler

Xin Yuan and Zhenhai Duan
Department of Computer Science
Florida State University
Tallahassee, FL 32306
Email: {xyuan, duan}@cs.fsu.edu

***Abstract*— In this paper, we propose an $O(1)$ complexity round robin scheduler, called Fair Round Robin (FRR), that provides good fairness and delay properties. Unlike existing $O(1)$ complexity round robin schedulers that can only achieve long term fairness, FRR not only provides proportional fairness, but also maintains a constant normalized worst-case fair index as defined in Bennett and Zhang's work. This means that FRR guarantees both short term and long term fairness among all backlogged flows.**

***Index Terms*— System design, Deterministic network calculus**

I. Introduction

Packet scheduling has been extensively studied in the last decade due to its importance in the provision of Quality of Service (QoS) guarantees in data networks. An ideal packet scheduler should have the following three properties. First, to be used in high speed networks, the scheduler should have a *low complexity*, preferably $O(1)$. Second, the scheduler should incur a *bounded delay* for each packet that reaches the head of the queue for a flow in order for the scheduler to support QoS guarantees. Last, the scheduler must provide *fairness* among the flows competing for the shared link so that each flow can get its fair share of the bandwidth.

While complexity and delay are well defined, the fairness of a scheduler needs further elaboration. Detailed discussion on this subject can be found in [8]. In this paper, we will use two well established fairness criteria, the *worst-case fairness* that was defined by Bennett and Zhang in [2] and the *proportional fairness* that was defined by Golestani in [5]. A scheduler, s, is worst-case fair to flow f_i if and only if the delay of a packet arriving at t on flow f_i is bounded by $\frac{Q_{i,s}(t)}{r_i} + C_{i,s}$, where $Q_{i,s}(t)$ is the queue size of f_i at t, r_i is the guaranteed rate of f_i, and $C_{i,s}$ is a constant independent of the queues of other flows. A scheduler is worst-case fair if it is worst-case fair to all flows in the system. If a scheduler, s, is worst-case fair, the fairness of the scheduler is measured by the *normalized worst-case fair index* [2]. Let R be the total link bandwidth. The normalized worst-case fair index for the scheduler, c_s, is defined as $c_s = \max_i\{\frac{r_i C_{i,s}}{R}\}$.

Let $S_{i,s}(t_1, t_2)$ be the amount of data of flow f_i sent during time period $[t_1, t_2)$. For any two flows f_i and f_j that are backlogged during any time period $[t_1, t_2)$, the proportional fairness requires the difference between the normalized service received by the two flows, $|\frac{S_{i,s}(t_1,t_2)}{r_i} - \frac{S_{j,s}(t_1,t_2)}{r_j}|$, to be bounded by a small constant.

Essentially, worst-case fairness requires the guaranteed rates of all flows in the system to be enforced at all time within a small error margin, while proportional fairness requires bandwidths allocated to any two backlogged flows to be proportional to their weights (guaranteed rates) at all time within a small error margin. To achieve both long term and short term fairness, a packet scheduler should provide both worst-case fairness and proportional fairness.

Existing scheduling algorithms can be classified into two types, timestamp based approaches [1], [2], [4], [5], [10] and round-robin algorithms [6], [7], [11], [15]. Timestamp based approaches have good bounded delay and fairness properties with a relatively high complexity, $O(log\ N)$, where N is the number of flows in the system. The implementation of timestamp based algorithms in high speed networks is problematic due to the logarithmic complexity. Round-robin algorithms have an $O(1)$ complexity, but in general do not have good bounded delay and fairness properties. For example, none of the existing round-robin schemes can achieve a constant normalized worst-case fair index.

In this paper, we propose a new packet scheduling scheme, called Fair Round-Robin (FRR). FRR combines the ideas in timestamp based approaches and round-robin algorithms. It maintains an O(1) per packet scheduling complexity in that the complexity is independent of the number of flows in the system. FRR has the desired properties of an ideal packet scheduler: an O(1) per packet processing complexity, a strict rate-proportional delay bound; and proportional and worst-case fairness for flows that are continuously backlogged.

The rest of the paper is structured as follows. Section II presents related work. Section III introduces the background of this work. Section IV describes FRR. Section V discusses the QoS properties of FRR. Section VI reports the results of the simulation study of FRR. In the study, FRR is compared with other representative packet scheduling algorithms including Weighted Fair Queueing [10], Smoothed Round Robin [6], and Stratified Round Robin [11]. Finally, Section VII concludes the paper.

This work is partially supported by NSF grants ANI-0106706, CCR-0208892, and CCF-0342540.

II. Related Work

Packet scheduling has been studied extensively and many scheduling algorithms have been proposed. Existing schedulers either use a timestamp based approach [1], [2], [4], [5], [10], [16], [19] or are round-robin algorithms [3], [6], [7], [11], [15].

Some timestamp based schedulers, such as Weighted Fair Queuing (WFQ) [10] and Worst-case Fair Weighted Fair Queuing (WF^2Q) [1], [2], closely approximate the Generalized Processor Sharing (GPS) [4], [10], which is an unrealistic algorithm that achieves perfect fairness and isolation among all flows. These schedulers compute a timestamp for each packet by emulating the progress of a reference GPS server and transmit packets in the increasing order of their timestamps. Both WFQ and WF^2Q have an $O(1)$ *GPS-relative delay* [17], that is, for each packet p, $F^p_{WFQ} - F^p_{GPS} \leq c_1 * \frac{L_M}{R}$ and $F^p_{WF^2Q} - F^p_{GPS} \leq c_2 * \frac{L_M}{R}$, where F^p_s is the time when packet p finishes service under scheduler s, L_M is the maximum packet size, R is the link bandwidth, and c_1 and c_2 are two constants. It has been shown in [17] that the lower bound computational complexity of any scheduling algorithm that guarantees an $O(1)$ GPS-relative delay bound is $\Omega(log\ N)$, where N is the number of flows in the system. Fortunately, having an $O(1)$ *GPS-relative delay* is not necessary for a scheduler to achieve worst-case and proportional fairness and, as will be shown in this paper, designing an $O(1)$ complexity scheduler that is worst-case and proportional fair is possible. Note that having an $O(1)$ *GPS-relative delay* is not sufficient for a scheduler to have a constant normalized worst-case fair index. For example, as shown in [2], WFQ does not have a constant normalized worst-case fair index.

Other timestamp based approaches, such as Self-Clocked Fair Queuing (SCFQ) [5] and Virtual Clock [19], compute timestamps without referring to a reference GPS server. These methods still need to sort packets according to their timestamps and still have an $O(log\ N)$ per packet processing complexity. The Leap Forward Virtual Clock [16] reduces the sorting complexity by coarsening timestamp values and has an $O(loglog\ N)$ complexity. This scheme requires complex data structures and is not suitable for hardware implementation.

Round-robin schedulers serve backlogged flows in some kind of round-robin fashion and have an $O(1)$ per packet processing complexity. To achieve fairness with different packet sizes, the Deficit Round Robin (DRR) scheme [15] augments the traditional round robin algorithm with a deficit counter and a quantum for each flow. A number of methods have recently been proposed to improve delay and burstiness properties of DRR [6], [7], [11]. The Smoothed Round Robin (SRR) scheme [6] improves the delay and burstiness properties by spreading the data of a flow to be transmitted in a round over the entire round using a weight spread sequence. Aliquem [7] allows the quantum of a flow to be scaled down, which results in better delay and burstiness properties. Both SRR and Aliquem improve the average packet delay over DRR, however, the worst-case single packet delay bound is proportional to the number of flows in the system. Like SRR, the Stratified Round Robin [11] scheme also tries to spread the data of a flow by using a deadline based scheduling scheme to ensure that all flows get their fair share of *slots*. Stratified Round Robin enjoys a single packet delay bound that is related to the guaranteed rate of the flow and is independent of the number of flows in the system. Bin Sort Fair Queuing (BSFQ) [3] uses an approximate bin sort mechanism to schedule packets. Each packet is assigned a timestamp (deadline). Packets are roughly sorted by placing them into bins according to their timestamps. No sorting of packets is performed within each bin. The worst-case single packet delay of BSFQ is proportional to the number of flows. Hybrid scheduling schemes [12], [13] have also been proposed, where the scheduling tasks are separated into two levels. While the algorithm components of these schemes are similar to those of FRR, the QoS properties of these schemes are not clear.

None of existing round-robin schemes has a constant normalized worst-cast fair index. To the best of our knowledge, our proposed scheme, FRR, is the first $O(1)$ time scheduler that has a constant normalized worst-cast fair index.

N	the number of flows in the system
n	the number of classes in the system
R	total link bandwidth
r_i	guaranteed bandwidth for flow f_i
$w_i = \frac{r_i}{R}$	the weight associated with flow f_i
l_M	maximum packet size
$S_{i,s}(t_1, t_2)$	the amount of work received by session i during $[t_1, t_2)$ under the s server
$S_{i,s}(t)$	the amount of work received by session i during $[0, t)$ under the s server
$F^k_{i,s}$	the departure time of the kth packet of flow f_i under the s server
F^p_s	the departure time of packet p under the s server
$Q_{i,s}(t)$	the queue size of flow f_i at time t under the s server
p^k_i	the kth packet on flow f_i

TABLE I
NOTATION USED IN THIS PAPER

III. Preliminaries

Major notations used in this paper are summarized in Table I. There are N flows f_1, f_2, ..., f_N sharing a link of bandwidth R. Each flow f_i has a minimum guaranteed rate of r_i. We will assume that $\sum_{i=1}^{N} r_i \leq R$. The weight w_i of flow f_i is defined as its guaranteed rate normalized with respect to the total rate of the link, i.e.,

$$w_i = \frac{r_i}{R}.$$

Thus, we have $\sum_{i=1}^{N} w_i \leq 1$. A scheduler determines the order of the packets to be served (transmitted), and thus, the bandwidth allocation for each flow. For example, using GPS, the bandwidth allocated to flow f_i is $\frac{w_i}{\sum_{f_j \text{ is backlogged}} w_j} R \geq r_i$.

A. Deficit Round Robin

Since the proposed scheduler, FRR, is built over Deficit Round Robin (DRR) [15], we will briefly describe DRR and present some properties of DRR that are needed to understand the properties of FRR.

Like the ordinary round robin scheme, DRR works in rounds. Within each round, each backlogged flow has an opportunity to send packets. Each flow f_i is associated with a quantity $quantum_i$ and a variable $deficitcounter_i$. The quantity $quantum_i$ is assigned based on the guaranteed rate for f_i and specifies the target amount of data that f_i should send in each round. However, since the scheduler operates in a packet-by-packet fashion, f_i may not be able to send exactly $quantum_i$ bytes in a round. The variable $deficitcounter_i$ is introduced to record the quantum that is not used in a round so that the unused quantum can be passed to the next round. To ensure that each flow can send at least one packet in a round, in this paper, we will assume that $quantum_i$ is larger than the maximum packet size, that is,

$$quantum_i \geq L_M.$$

More details about DRR can be found in [15]. Some properties of DRR are summarized in the following lemmas. Due to the page limit constraint, we omit the proofs of these lemmas and some other lemmas and theorems in the later of this paper. The proofs of all lemmas and theorems are available online in the technical report version of this paper [18].

Lemma 1: Assuming that flow f_i is continuously backlogged during $[t_1, t_2)$. Let X be the smallest number of continuous DRR rounds that completely enclose $[t_1, t_2)$. The service received by f_i during this period, $S_{i,DRR}(t_1, t_2)$, is given by

$$(X-3)quantum_i \leq S_{i,DRR}(t_1, t_2) \leq (X+1)quantum_i.$$

Proof: See [18]. □

Lemma 2: Let f_1, ..., f_N be the N flows in the system with guaranteed rates r_1, ...,r_N. $\sum_{i=1}^{N} r_i \leq R$. Let $r_{min} = \min_i\{r_i\}$ and $r_{max} = \max_i\{r_i\}$. Assume that there exists a constant C such that $r_{max} \leq C * r_{min}$, and that DRR is used to schedule the flows with $quantum_i = L_M * \frac{r_i}{r_{min}}$. The following statements are true.

1) Let packet p arrive at the head of the queue for f_i at time t. There exists a constant c_1 such that packet p will be serviced before $t + c_1 \times \frac{L_M}{r_i}$.
2) The normalized worst-case fair index of DRR in such a system is a constant.
3) Let f_i and f_j be continuously backlogged during any given time period $[t_1, t_2)$, there exists two constants c_1 and c_2 such that the normalized service received by the two flows during this period is bounded by

$$|\frac{S_{i,DRR}(t_1, t_2)}{r_i} - \frac{S_{j,DRR}(t_1, t_2)}{r_j}| \leq c_1\frac{L_M}{r_i} + c_2\frac{L_M}{r_j}$$

Proof: See [18]. □

Lemma 2 shows that when the weights of the flows in the system are similar (within a constant factor), DRR has the following three properties. First, the worst-case single packet delay depends on the guaranteed rate for the flow and is independent of the number of flows in the system. Second, DRR has a constant normalized worst-case fair index, which means that DRR can guarantee short term worst-case bandwidth to all flows. Third, DRR provides proportional fairness. Thus, DRR is an excellent scheduler when the weights of flows are similar. The problem with DRR is that when the weights of the flows differ significantly, which is common in practice, the three properties do not hold simultaneously. In particular, with DRR, flows with large weights can be significantly affected by many flows with small weights both in terms of packet delay and short term bandwidth allocation.

Fair Round Robin (FRR), our proposed scheduler, extends DRR such that the three quality of service properties hold for any weight distribution, while maintaining an $O(1)$ complexity. The basic idea is as follows. FRR groups flows with similar weights into classes and uses a variation of DRR to schedule packets within each class. As shown in Lemma 2, DRR can achieve good QoS properties for flows in each class. Thus, the challenge is to isolate the classes so that flows in different classes, which are flows with significantly different weights, do not affect each other too much. FRR uses a timestamp based scheduler to isolate the classes. As a result, FRR schedules packets in two levels, a timestamp based inter-class scheduling and a DRR based intra-class scheduling. Since multiple flows are grouped in each class, the weight assigned to a class may change dynamically to reflect the number of active flows in the class. Traditional timestamp based approaches either cannot be directly applied or do not support provable QoS properties when weights of flows can change dynamically. A new timestamp based scheduler is developed to closely approximate GPS with dynamically changing weights.

IV. FRR: A Fair Round Robin Scheduler

In this section, we will describe FRR, a proportional and worst-case fair round robin scheduler. Like the stratified round robin scheme [11], FRR groups flows into a number of classes with each class containing flows with similar weights. For $k \geq 1$, class F_k is defined as

$$F_k = \{f_i : \frac{1}{C^k} \leq w_i < \frac{1}{C^{k-1}}\},$$

where C is a constant. Let r be the smallest unit of bandwidth that can be allocated to a flow. The number of classes is $n = \lceil log_C(\frac{R}{r})\rceil$. Consider an example where $R = 1Tbps$, $r = 1kbps$, $C = 8$. $n = \lceil log_8(10^9)\rceil = 10$. Thus, only 10 classes are needed for this case. When $C = 2$, $n = \lceil log_2(10^9)\rceil = 30$. These examples show that the number of classes to be maintained is small in practical cases. In the rest of this paper, we will assume that the number of classes, n, is a small constant. Notice that n is related only to R, r, and C and is independent of N. Thus, even in theory, n **is an** $O(1)$ **constant with respect to** N, the number of flows in system.

FRR has two scheduling components, an intra-class scheduling that determines the order of the packets within each

class and the weight of the class, and an inter-class scheduling that determines the class, and thus, the packet within the class, to be transmitted over the link. Next, we will describe these two components.

A. Inter-class scheduling

As discussed earlier, the inter-class scheduler, which is designed to isolate classes, is a timestamp based scheduler. Since multiple flows are grouped into each class, the rate allocated to a class may need to be changed at different times so that each flow within a class can have its fair share of bandwidth. Thus, the inter-class scheduler should be able to handle the dynamically changing weights and achieve fair sharing of bandwidth in the presence of the dynamically changing weights. Note that while GPS achieves fair sharing of bandwidth when the weights of classes change dynamically, none of the existing timestamp based schemes, which closely approximate GPS when the weights of flows do not change, can closely approximate GPS when the weights change dynamically. We develop a new scheme called Dynamic Weight Worst-case Fair weighted Fair Queuing (DW^2F^2Q). DW^2F^2Q has the same scheduling result as WF^2Q [2] when the weights do not change. Theorems presented later in this section show that, when the number of classes is a small constant, DW^2F^2Q closely approximates GPS with dynamic weights. Next, we will first discuss how to keep track of GPS progress when the weights change dynamically and then describe how packets are scheduled by DW^2F^2Q.

Let us denote an event in the system the following: (1) the arrival of a packet to the GPS server, (2) the departure of a packet from the GPS server, and (3) the weight change of a class. Let t_j be the time at which the jth event occurs. Let the time of the first arrival of a busy period be denoted as $t_1 = 0$. For each $j = 2, 3, ...$, the set of classes that are busy in the interval $[t_{j-1}, t_j)$ is denoted as B_{j-1}. Let us denote $w_{i,j-1}$ the weight for class i during the interval $[t_{j-1}, t_j)$, which is a fixed value. The GPS progress of class i during the time interval $[t_{j-1}, t_{j-1} + \tau)$, where $0 < \tau \leq t_j - t_{j-1}$, is

$$S_{i,GPS}(t_1) = 0$$

$$S_{i,GPS}(t_{j-1} + \tau) = S_{i,GPS}(t_{j-1}) + \frac{w_{i,j-1}}{\sum_{k \in B_{j-1}} w_{k,j-1}} \times R \times \tau$$

DW^2F^2Q keeps track of the GPS progress of all the classes using the above formula. Notice that for each event, the GPS progress of all classes may need to be updated. The per event computational complexity is $O(n)$, where n is the number of classes, which is a small *constant* in FRR. Thus, assuming that weight change is less frequent than packet arrival, which is true in FRR, the per packet computational complexity for maintaining GPS progress is $O(n) = O(1)$. Since DW^2F^2Q schedules packets only at packet boundaries (packet arrivals and departures), it is sufficient to maintain accurate GPS progress at packet boundaries.

In addition to keeping track of GPS progress, DW^2F^2Q also records the amount of data of each class that have been serviced. Assume that the server needs to decide the next packet at time t_j. Let $size_i(t_j)$ be the size of the packet at the head of class i at time t_j, $S_{i,GPS}(t_j)$ be the amount of data of session i served under GPS, and $S_{i,DW^2F^2Q}(t_j)$ be the amount of data session i actually served under DW^2F^2Q. The head of class i is scheduled at t_j if and only if the following two conditions are met:

- Condition 1): $S_{i,GPS}(t_j) \geq S_{i,DW^2F^2Q}(t_j)$.
- Condition 2): Let $nf(i)$ be the estimated GPS finishing time of the head packet of class i, p_i^k. For a backlogged class i, $nf(i)$ is computed as follows. If $size_i(t_j) + S_{i,DW^2F^2Q}(t_j) > S_{i,GPS}(t_j)$,

$$nf(i) = t_j + \frac{size_i(t_j) + S_{i,DW^2F^2Q}(t_j) - S_{i,GPS}(t_j)}{R \times w_{i,j}}.$$

If $size_i(t_j) + S_{i,DW^2F^2Q}(t_j) \leq S_{i,GPS}(t_j)$,

$$nf(i) = F_{i,GPS}^k.$$

Class i has the smallest estimated GPS finishing time, $nf(i)$, among all backlogged classes. For classes with the same $nf(i)$, the class number can be used to break the tie.

The first condition enforces that DW^2F^2Q does not schedule a packet before the GPS starting time of the packet. This ensures that DW^2F^2Q can be at most one packet ahead of GPS. The second condition enforces that packets are ordered based on the estimated GPS finishing time. The class whose head packet has the smallest estimated GPS finishing time is scheduled. In estimating the GPS finishing time, there are two cases. The first case is when the packet has not departed under GPS ($size_i(t_j) + S_{i,DW^2F^2Q}(t_j) > S_{i,GPS}(t_j)$). In this case, the finishing time is estimated using the current weight, $w_{i,j}$. $R \times w_{i,j}$ is the GPS guaranteed rate for weight $w_{i,j}$. Note that since the weights of classes can change dynamically, this estimated GPS finishing time of a packet may be inaccurate. The second case is when the packet has departed under GPS ($size_i(t_j) + S_{i,DW^2F^2Q}(t_j) \leq S_{i,GPS}(t_j)$). In this case, the actual GPS departure time, which is the accurate GPS finishing time, is used as the timestamp. Hence, DW^2F^2Q uses accurate information to schedule packets that fall behind GPS and may use inaccurate information to schedule packets that are ahead of GPS. The complexity to schedule a packet is $O(n)$, where n is the number of classes, which is a small *constant* in FRR. Thus, the per-packet computational complexity of the inter-class scheduling is $O(n) = O(1)$. The following sequence of theorems shows some properties of DW^2F^2Q.

Theorem 1: DW^2F^2Q is work conserving.

Proof: Since GPS is work-conserving, we will prove the theorem by showing that DW^2F^2Q has the same idle and busy periods as GPS. Assuming that DW^2F^2Q and GPS have different idle and busy periods. Let t be the first occurrence when GPS and DW^2F^2Q are not in the same state. There are two cases.

Case 1: GPS is idle and DW^2F^2Q is busy, serving packet p. Since t is the first occurrence when GPS and DW^2F^2Q are not in the same state, the amount of data served during $[0, t)$ must be the same for the two scheduling schemes. Since

p is currently being served under DW^2F^2Q, from condition 1), p must be started before t under GPS. Since GPS is idle at time t, packet p must finish before t under GPS. Hence, there must exist a packet q such that q has not been served under GPS during $[0,t)$ and has been served by DW^2F^2Q during $[0,t)$. Since GPS is idle at t, packet q should start after t under GPS, which indicates that q cannot be served under DW^2F^2Q during $[0,t)$ since condition 1) does not hold. This is the contradiction.

Case 2: GPS is busy and DW^2F^2Q is idle. Let packets p_1, p_2, ..., p_i be the packets departed under GPS during $[0,t)$ and packets $cp_1, ..., cp_j$ be the packets currently in progress under GPS. Since GPS is busy, at least one packet is being serviced at time t. Since DW^2F^2Q is idle at t, all packets satisfy condition 1) should have been served, that is, packets p_1, p_2, ..., p_i and $cp_1, ..., cp_j$ are all served during $[0,t)$ under DW^2F^2Q. Thus, during $[0,t)$, DW^2F^2Q sends more data than GPS and t cannot be the first occurrence that GPS and DW^2F^2Q are not in the same state. □

Since both GPS and DW^2F^2Q are work-conserving disciplines, their busy periods coincide. In the rest of the section, we will consider packet scheduling within one busy period. Let $F^k_{i,s}$ be the departure time of the kth packet in class i under server s.

Lemma 3: If $F^k_{i,GPS} \leq F^m_{j,GPS}$, $F^k_{i,DW^2F^2Q} < F^{m+1}_{j,DW^2F^2Q}$.

Proof: Let p^l_i be the packet at the head of class i at time t when p^{m+1}_j is at the head of class j and is eligible to be transmitted. From Condition 1), $t \geq F^m_{j,GPS}$ and the estimate finishing time for packet p^{m+1}_{j,DW^2F^2Q}, $nf(j) > F^m_{j,GPS}$.

If $l > k$, we have $F^k_{i,DW^2F^2Q} < F^{m+1}_{j,DW^2F^2Q}$ and the lemma is proved. If $l \leq k$,

$$F^l_{i,GPS} < F^{l+1}_{i,GPS} < ... < F^k_{i,GPS} \leq F^m_{j,GPS} < nf(j).$$

Since packets p^l_i, p^{l+1}_i,..., p^k_i have departed before time t under GPS, the actual GPS departing times of the packets will be assigned as the estimated GPS finishing times of these packets. Thus, all these packets will have a smaller estimated GPS finishing time than packet p^{m+1}_j. Since DW^2F^2Q selects the class with the smallest estimated GPS finishing time for transmission, all these packets will be transmitted before p^{m+1}_j. Thus, $F^k_{i,DW^2F^2Q} < F^{m+1}_{j,DW^2F^2Q}$. □

By allowing classes to change weights dynamically, DW^2F^2Q may not be able to estimate the GPS departure time accurately since it cannot predict the future weight changes. However, Lemma 3 indicates that DW^2F^2Q can at most introduce one packet difference between any two classes in comparison to GPS. This leads to the following theory that states relation of GPS departure time and DW^2F^2Q departure time. Let F^p_s be the time packet p departs under server s.

Theorem 2: Let n be the number of classes in the system,

$$F^p_{DW^2F^2Q} - F^p_{GPS} \leq (n-1)\frac{L_M}{R}.$$

Proof: Consider any busy period and let the time that it begins be time zero. Let p_k be the kth packet of size s_k in the busy period to depart under GPS. We have

$$F^{p_k}_{GPS} \geq \frac{s_1 + s_2 + ... + s_k}{R}$$

Now consider the departure time of p_k under DW^2F^2Q. From Lemma 3, each class can have at most one packet whose GPS finishing time is after packet p_k and whose DW^2F^2Q finishing time is before packet p_k. Hence, there are at most $n-1$ packets (from the $n-1$ other classes) that depart before packet p_k under DW^2F^2Q and have a GPS finishing time after $F^{p_k}_{GPS}$. Let the $n-1$ packets be e_1, e_2, ..., e_{n-1} with sizes se_1, se_2, ..., se_{n-1}. All other packets depart before p^k under DW^2F^2Q must have GPS finishing times earlier than $F^{p_k}_{GPS}$. We have

$$F^{p_k}_{DW^2F^2Q} \leq \frac{s_1 + ... + s_k + se_1 + ... + se_{n-1}}{R}$$

Thus,

$$F^{p_k}_{DW^2F^2Q} - F^{p_k}_{GPS} \leq (n-1)\frac{L_M}{R}.$$

□

Theorem 2 gives the relative delay bound between GPS and DW^2F^2Q. The bound is proportional to the number of classes. Although this bound is in general not a good bound, when the number of classes in the system is a constant, as in FRR, this bound is sufficient: the packet departure times under GPS and DW^2F^2Q differ by at most a constant number of packets.

Theorem 3: For all time τ and class i,

$$S_{i,GPS}(0,\tau) - S_{i,DW^2F^2Q}(0,\tau) \leq (n-1) * L_M.$$

Proof: See [18]. □

Theorem 4: For all time τ and class i,

$$S_{i,DW^2F^2Q}(0,\tau) - S_{i,GPS}(0,\tau) \leq L_M.$$

Proof: Straightforward from Condition 1) of the DW^2F^2Q scheduling scheme. □

These theorems establish that DW^2F^2Q closely approximates GPS for dynamic weights when the number of classes is a constant: the difference between the total service given to a particular class for the two scheduling schemes is within a constant number of packets. Notice that one key point that makes DW^2F^2Q work in FRR is that the number of classes in FRR is a small constant.

B. Intra-class scheduling

Since the inter-classes scheduling scheme, DW^2F^2Q, schedule packets based on the weights of the classes, and since many flows are grouped into each class, intra-class scheduling needs to decide (1) the weights used to send packets in a class, and (2) the order of packets coming from different flows. Note that intra-class scheduling does not decide the actual time a packet is serviced. This function is performed by inter-class scheduling.

As discussed earlier, DRR offers good QoS properties when it is used to schedule flows with similar weights. However, naively applying DRR in the intra-class scheduling

variable	explanation
$deficitcount_i$	the deficit count for flow f_i
$remaindeficit$	the sum of quantum not used in the DRR round
$lastingflowlist$	the flows that last to the next frame
$framesize$	the size of the frame
$frameweight$	the weight for the frame
$remainsize$	size of the part of a packet that belongs to current frame

TABLE II

MAJOR VARIABLES USED IN THE FRAME CALCULATION ALGORITHM

does not yield a fair scheduler even assuming that the inter-class scheduler is GPS. In order to obtain a fair scheduler, the intra-class scheduler must be able to transfer the fairness at the class level (provided by DW^2F^2Q) to the fairness at the flow level. In FRR, intra-class scheduling uses a frame based approach. The packet stream within a class is partitioned into logical frames with packets in each frame being scheduled using the same weight. The intra-class scheduling scheme is called *Lookahead Deficit Round Robin with Weight Adjustment* ($LDRRWA$), a variation of DRR. In $LDRRWA$, backlogged flows are served in a round-robin fashion. To offset the weight differences among the flows in a class, each flow $f_i \in F_k = \{f_i : \frac{1}{C^k} \leq w_i < \frac{1}{C^{k-1}}\}$ is assigned a quantum of

$$quantum_i = C^k w_i L_M.$$

Since $\frac{1}{C^k} \leq w_i < \frac{1}{C^{k-1}}$,

$$L_M \leq quantum_i < C \times L_M.$$

A $LDRRWA$ frame is related to, but different from, a DRR round. A $LDRRWA$ frame is basically a DRR round plus some packets that are in the next DRR round and are moved into the current frame by the *lookahead operation*. Each $LDRRWA$ frame, together with its associated weight, is computed using the algorithm shown in Fig. 1. The major variables used in the algorithm are summarized in Table II. Like DRR, variable $deficitcount_i$ is associated with flow f_i to maintain the credits to be passed over to the next DRR round and decide the amount of data to be sent in one round. After each DRR round, $remaindeficit$ maintains the sum of the quanta not used in the current DRR round, that is, the quanta that cannot be used since the size of the next backlogged packet is larger than the remaining quanta for a flow. In traditional DRR, these unused quanta will be passed to the next DRR round. In $LDRRWA$, in addition to passing the unused quanta to the next DRR round, some packets that would be sent in the next DRR round are placed in the current $LDRRWA$ frame so that at frame boundaries $remaindeficit$ is always equal to 0. This is the lookahead operation. As will be shown later, this lookahead operation is the key to ensure that each frame is properly sized such that long lasting flows can get their fair share of the bandwidth. The $lastingflowlist$ contains the list of flows that are backlogged at the end of the current DRR round. Flows in $lastingflowlist$ are candidates

Algorithm for computing the next frame for class F_k

(1) $remaindeficit = framesize = 0$
(2) $lastingflowlist = NULL$
(3) **if** $(remainsize > 0)$ **then**
/*The partial packet belongs to this frame */
(4) $framesize = framesize + remainsize$
(5) **end if**
/* forming the DRR round */
(6) **for** each active flow f_i **do**
(7) $deficitcount_i = deficitcount_i + quantum_i$
(8) **while** $(deficitcount_i > 0)$ **and** (f_i not empty) **do**
(9) $pktsize = size(head(f_i))$
(10) **if** $(pktsize < deficitcount_i)$ **then**
(11) remove head from f_i and put it in the frame
(12) $framesize = framesize + pktsize$
(13) $deficitcount_i = deficitcount_i - pktsize$
(14) **else** break
(15) **end if**
(16) **end while**
(17) **if** (f_i is empty) **then**
(18) $deficitcount_i = 0$
(19) **else**
(20) $remaindeficit = remaindeficit + deficitcount_i$
(21) insert f_i to $lastingflowlist$
(22) **end if**
(23) **end for**
/* lookahead operation */
(24) f_i = head(lastingflowlist)
(25) **while** $(f_i \neq NULL)$ **and** $(remaindeficit > 0)$ **do**
(26) $pktsize = size(head(f_i))$
(27) **if** $(pktsize < remaindeficit)$ **then**
(28) remove head from f_i and put it in the frame
(29) $framesize = framesize + pktsize$
(30) $remaindeficit = remaindeficit - pktsize$
(31) $deficitcount_i = deficitcount_i - pktsize$
(32) **else** break
(33) **end if**
(34) $f_i = nextflow(f_i)$
(35) **end while**
(36) **if** $(f_i \neq NULL)$ **then**
(37) $pktsize = size(head(f_i))$
(38) remove head from f_i and put it in the frame
(39) $framesize = framesize + remaindeficit$
(40) $remainsize = pktsize - remaindeficit$
(41) $deficitcount_i = deficitcount_i - pktsize$
(42) **end if**
/* computing the weight */
(43) $weight = totalquantum = 0$
(44) **for** each active flow f_i in current frame **do**
(45) $weight = weight + w_i$
(46) $totalquantum = totalquantum + quantum_i$
(47) **end for**
(48) $frameweight = weight * \frac{framesize}{totalquantum}$
(49) **if** $(frameweight < \frac{1}{C^k})$ $frameweight = \frac{1}{C^k}$

Fig. 1. The algorithm for computing the next frame for class F_k

to supply packets for the lookahead operation. $Frameweight$ is the weight to be used by inter-class scheduling for the current frame. Variable $framesize$ records the size of the current frame. FRR needs this information to determine when to invoke the frame computation algorithm to compute the next frame (and thus, when to change weights for classes). Note that frame boundaries may not align with packet boundaries since FRR needs to enforce that $remaindeficit = 0$ at frame boundaries. Thus, a packet may belong to two frames (in the simulated GPS, weight change may happen within a packet). Variable $remainsize$ is the size of the part of the last packet in the frame that belongs to the next frame, and thus, should be counted in the $framesize$ for the next frame. In FRR, frame is a logical concept that affects only the progress of the simulated GPS. Thus, not aligning frame boundaries with packet boundaries does not cause problems in the actual packet scheduling.

Let us now examine the algorithm in Fig. 1. In the initialization phase, line (1) to line (5), variables are initialized and $remainsize$ is added to $framesize$, which effectively includes the partial packet in the frame to be computed. After the initialization, there are three main components in the algorithm: forming a DRR round, lookahead operation, and weight calculation. In the first component, line (6) to line (23), the algorithm puts all packets in the current DRR round that have not been served into the current frame. In the second component, line (24) to line (42), the algorithm performs the lookahead operation by moving some packets in the next DRR round into the current frame so that $remaindeficit = 0$ at the frame boundary. This is done by allowing some flows to borrow credits from the next DRR round. Since $remaindeficit = 0$, no credit is passed from one frame to the next frame for the class that aggregates many flows. Notice that each backlogged flow can contribute at most one packet in the lookahead operation. Notice also that a class as a whole does not pass credits between frames. However, for an individual flow, credits may still pass from one frame to the next. As a result, the $deficitcount_i$ variable may have a negative or positive value at frame boundaries.

The last component in the algorithm, line (43) to line (49) calculates the weight for the frame. As will be proved in the following lemmas, the weights are assigned such that (1) the weight for the frame is always less than or equal to the sum of all weights of the active flows in the frame (Lemma 5) and (2) the service time for each frame for class F_k is at most $\frac{C^k L_M}{R}$ (Lemma 6). Notice that $\frac{C^k L_M}{R} = \frac{C^k * w_i * L_M}{w_i * R} = \frac{quantum_i}{r_i}$. $\frac{C^k L_M}{R}$ is the "standard" time for a flow in F_k to send its quantum using its guaranteed rate.

The complexity of the algorithm is $O(M)$, where M is the number of packets in a frame. Hence, the amortised per packet complexity for frame construction is $O(1)$. This algorithm is invoked in two occasions: (1) when the class becomes backlogged (when a packet arrives at an idle class), and (2) when the current frame is finished under the simulated GPS. To invoke the algorithm at the time when the current frame is finished under the simulated GPS, a timer is associated with each class. The timers record the estimated GPS finishing times of the current frames. Every time a frame departs or arrives under GPS, the timers for all classes are updated to reflect the changing of the GPS progress and the changing of weights. When a timer expires, the corresponding frame is finished under GPS and the algorithm is invoked to compute a new frame. Maintaining the timers results in $O(n) = O(1)$ per frame overhead. Next, we will prove a sequence of lemmas that show the properties of $LDDRWA$.

Lemma 4: Assuming that flow f_i is continuously backlogged during $[t_1, t_2)$. Let X be the smallest number of continuous $LDRRWA$ frames that completely enclose $[t_1, t_2)$. The service received by f_i during this period, denoted as $S_{i,LDRRWA}(t_1, t_2)$, is given by

$$(X-4)quantum_i \leq S_{i,LDRRWA}(t_1, t_2) \leq (X+2)quantum_i.$$

Proof: The notation $S_{i,LDRRWA}(t_1, t_2)$ is abused in this lemma since $LDRRWA$ does not decide the actual timing to service packets. In this lemma, $S_{i,LDRRWA}(t_1, t_2)$ denotes the amount of data for a continuously backlogged flow f_i in X continuous $LDRRWA$ frames (of a particular class) using any inter-class scheduling scheme.

Since f_i is continuously backlogged, it will try to send as many packets as possible in each frame. Since X frames enclose $[t_1, t_2)$, flow f_i will fully utilize at least $X-2$ frames (all but the first frame and the last frame). In the $X-2$ frames, $(X-2) \times quantum_i$ credits are generated for flow f_i. The lookahead operation in the frame prior to the $X-2$ frames may borrow at most one packet, whose size is less than L_M, from f_i in the first of the $X-2$ frames and flow f_i in the last of the $X-2$ frames may pass at most L_M credits to the next frame. Note that the lookahead operation borrows at most one packet from each backlogged flow. Note also that, while frames for a class do not pass credits between each other accumulatively for all flows in the class, an individual flow may pass credits across frame boundaries. Thus,

$$S_{i,LDRRWA}(t_1, t_2) \geq (X-2) \times quantum_i - L_M - L_M.$$

Since $quantum_i \geq L_M$,

$$S_{i,LDRRWA}(t_1, t_2) \geq (X-4) \times quantum_i.$$

On the other hand, f_i will be serviced in at most all the X frames, which produces $X \times quantum_i$ credits for f_i during this period of time. Flow f_i in the frame prior to the X frames may have at most L_M left-over credits and the lookahead operation in the last of the X frames may borrow at most L_M credits from f_i in the next frame. Thus,

$$\begin{aligned} S_{i,LDRRWA}(t_1, t_2) &\leq X \times quantum_i + L_M + L_M \\ &\leq (X+2)quantum_i. \end{aligned}$$

□

Comparing Lemma 4 and Lemma 1, we can see the similarity between DRR and $LDRRWA$. However, as we will show in the following lemmas, unlike DRR rounds, $LDRRWA$

frames are always properly sized so that continuously backlogged flows can obtain their fair shares of the bandwidth.

Lemma 5: The weight for a frame is less than or equal to the sum of the weights of the active flows in the frame.

Proof: Since no credit is passed between frames, the amount of data that is allowed to send in one frame is at most the total quanta of all active flows generated in that frame. Thus, $framesize \leq \sum_{f_i} quantum_i$. From line (48) in Figure 1, $frameweight = \sum_{f_i} w_i * \frac{framesize}{\sum_{f_i} quantum_i} \leq \sum_{f_i} w_i$. If the condition in line (49) is true, $frameweight$ is assigned to the minimum weight of a flow in the class and thus the same conclusion holds. □

For any given time, let cw_i, $1 \leq i \leq n$ be the weights for the n classes (cw_i may change over time). Lemma 5 establishes that $\sum_{i=1}^{n} cw_i \leq \sum_{i=1}^{N} w_i \leq 1$. Thus, under GPS (with dynamic weights), the bandwidth allocated to class i is given by

$$\frac{cw_i}{\sum_{i=1}^{n} cw_i} R \geq R \times cw_i.$$

We will call $R \times cw_i$ the GPS *guaranteed rate* since this rate is guaranteed when the weight for a class is cw_i *regardless how other classes change weights*.

Lemma 6: Under GPS, the time to service each $LDRRWA$ frame in class F_k is at most $C^k \frac{L_M}{R}$.

Proof: Consider first that the condition in line (49) is not true and the frame weight is computed in line (48) in Fig. 1. In this case,

$$\begin{aligned} frameweight &= \sum_{f_i} w_i * \frac{framesize}{\sum_{f_i} quantum_i} \\ &= \sum_{f_i} w_i * \frac{framesize}{\sum_{f_i} C^k w_i L_M} = \frac{framesize}{C^k L_M} \end{aligned}$$

Thus, the GPS guaranteed rate for this class is $R\frac{framesize}{C^k L_M}$ and the total time to serve this frame is at most

$$\frac{framesize}{R\frac{framesize}{C^k L_M}} = \frac{C^k L_M}{R}.$$

If the condition in line (49) is true, the weight for the frame is increased and the conclusion still holds. □

Lemma 7: Under GPS, the time to service X bytes of data in the queue for class F_k is at most $\frac{XC^k}{R}$.

Proof: The minimum weight assigned to a backlogged class F_k is $\frac{1}{C^k}$. Thus, the GPS guaranteed rate for class F_k is at least $\frac{R}{C^k}$. Thus, the time to serve a queue of size X bytes in class F_k is at most $\frac{X}{\frac{R}{C^k}} = \frac{XC^k}{R}$. □

Lemma 8: For a class F_k frame of size no smaller than L_M, the service time for the frame is exactly $C^k \frac{L_M}{R}$ using the GPS guaranteed rate.

Proof: When the frame is larger than L_M, $frameweight = \frac{framesize}{C^k L_M} \geq \frac{1}{C^k}$. Thus, the GPS guaranteed rate for the frame is $R\frac{framesize}{C^k L_M}$ and the service time for the frame with the guaranteed rate is

$$\frac{framesize}{R\frac{framesize}{C^k L_M}} = \frac{C^k L_M}{R}.$$

□

Lemma 9: Let a class F_k frame contains packets of a continuously backlogged flow f_i, the size of frame is no smaller than L_M.

Proof: Straight-forward from the fact that no credit is passed from the previous frame and to the next frame and that $quantum_i \geq L_M$. □

Lemma 10: Let $f_i \in F_k$ and $f_j \in F_m$ be continuously backlogged during $[t_1, t_2)$. $k \geq m$. Let X_k and X_m be the smallest numbers of F_k and F_m frames that completely enclose $[t_1, t_2)$. Assume that classes F_k and F_m are serviced with the GPS guaranteed rate.

$$(X_k - 1)C^{k-m} \leq X_m \leq X_k C^{k-m} + 1.$$

Proof: Since $f_i \in F_k$ and $f_j \in F_m$ are continuously backlogged during $[t_1, t_2)$, the sizes of all frames during this period are no smaller than L_M (Lemma 9). From Lemma 8, using the GPS guaranteed rate, the time to service a class F_k frame is exactly $\frac{C^k L_M}{R}$ and the time for a class F_m frame is exactly $\frac{C^m L_M}{R}$. Since X_k and X_m are the smallest numbers of F_k and F_m frames that completely enclose $[t_1, t_2)$, we have

$$t2 - t1 \leq X_k \frac{C^k L_M}{R} \leq t2 - t1 + \frac{C^k L_M}{R}$$

$$t2 - t1 \leq X_m \frac{C^m L_M}{R} \leq t2 - t1 + \frac{C^m L_M}{R}.$$

Hence, $(X_k - 1)C^{k-m} \leq X_m \leq X_k C^{k-m} + 1$. □

Lemma 11 relaxes the condition in Lemma 10 by not requiring each class to be serviced with its GPS guaranteed rate.

Lemma 11: Let $f_i \in F_k$ and $f_j \in F_m$ be continuously backlogged during $[t_1, t_2)$. $k \geq m$. Let X_k and X_m be the smallest numbers of F_k and F_m frames that completely enclose $[t_1, t_2)$. Assume that the inter-class scheduler is GPS.

$$(X_k - 1)C^{k-m} \leq X_m \leq X_k C^{k-m} + 1.$$

Proof: This lemma relaxes the condition in Lemma 10 by not requiring each class to be serviced with its GPS guaranteed rate. Since $f_i \in F_k$ and $f_j \in F_m$ be continuously backlogged during $[t_1, t_2)$, the sizes of all frames during this period are no smaller than L_M (Lemma 9). Let us partition the duration $[t_1, t_2)$ into smaller intervals $[a_1 = t_1, b_1)$, $[a_2 = b_1, b_2)$, ..., $[a_Y = b_{Y-1}, b_Y = t_2)$ such that within each interval $[a_h, b_h)$, $1 \leq h \leq Y$, the weights of all classes are fixed. Let F_1, ..., F_n be the n classes in the system. Let class F_k have weight w_k^h during interval $[a_h, b_h)$, $1 \leq h \leq Y$ (If F_k is not backlogged, $w_k^h = 0$). The amount of class F_k data sent during $[a_h, b_h)$ is thus,

$$\frac{w_k^h}{\sum_{i=1}^{n} w_i^h} R * (b_h - a_h).$$

Consider a reference scheduling system that contains three classes RF_k, RF_m, and RF_o. Let us use intervals $[aa_1 = t_1, bb_1)$, $[aa_2 = bb_1, bb_2)$, ..., $[aa_Y = bb_{Y-1}, bb_Y)$ to emulate the behavior of classes F_k and F_m during intervals $[a_1 = t_1, b_1)$, $[a_2 = b_1, b_2)$, ..., $[a_Y = b_{Y-1}, b_Y)$ respectively. Let rw_k^h be the weight for class RF_k during interval $[aa_h, bb_h)$,

$1 \leq h \leq Y$. Let rw_m^h be the weight for class RF_m during interval $[aa_h, bb_h)$, $1 \leq h \leq Y$. Let rw_o^h be the weight for class RF_o during interval $[aa_h, bb_h)$, $1 \leq h \leq Y$. The weights and the duration of each interval are given as follows:

$$rw_k^h = w_k^h, rw_m^h = w_m^h, rw_o^h = 1 - w_k^h - w_m^h, 1 \leq h \leq Y$$

and

$$bb_h = aa_h + \frac{b_h - a_h}{\sum_{i=1}^{n} w_i^h}, 1 \leq h \leq Y.$$

It can be verified that the amount of classes RF_k and RF_m data sent in an interval $[aa_h, bb_h)$, $1 \leq h \leq Y$, is exactly the same as the amount of classes F_k and F_m data sent in an interval $[a_h, b_h)$, $1 \leq h \leq Y$, respectively. In an interval $[aa_h, bb_h)$, $1 \leq h \leq Y$, let us further assume that Class RF_k has exactly the same sequence of packets as Class F_k has in interval $[a_h, b_h)$ and that Class RF_m has exactly the same sequence of packets as Class F_m has in interval $[a_h, b_h)$. The progress of classes F_k and F_m during $[t_1, t_2)$ is exactly the same as the progress of class RF_k and RF_m during $[aa_1, bb_Y)$

In the reference system, classes RF_m and RF_k are serviced with the GPS guaranteed rate during $[aa_1, bb_Y)$. Let RX_k and RX_m be the smallest numbers of RF_k and RF_m frames that completely enclose $[aa_1, bb_Y)$. From Lemma 10,

$$(RX_k - 1)C^{k-m} \leq RX_m \leq RX_k C^{k-m} + 1.$$

Let X_k and X_m be the smallest number of F_k and F_m frames that completely enclose $[t_1, t_2)$. Since the progress of classes F_k and F_m during $[t_1, t_2)$ is exactly the same as the progress of class RF_k and RF_m during $[aa_1, bb_Y)$, we have $X_k = RX_k$ and $X_m = RX_m$. Thus,

$$(X_k - 1)C^{k-m} \leq X_m \leq X_k C^{k-m} + 1.$$

□

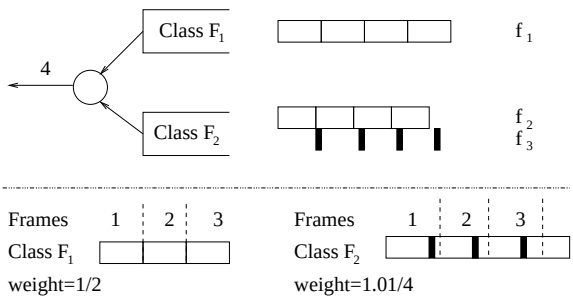

Fig. 2. An example

In the following, we will use an example to illustrate how $LDRRWA$ interacts with inter-class scheduling and how it transfers the fairness at the class level to the flow level. To simplify the discussion, we will assume that GPS is the inter-class scheduling algorithm. Consider scheduling for a link with 4 units of bandwidth with the following settings. $C = 2$ and there are two classes where $F_1 = \{f_i : \frac{1}{2} \leq w_i < 1\}$ and $F_2 = \{f_i : \frac{1}{4} \leq w_i < \frac{1}{2}\}$. Three flows, f_1, f_2 and f_3, with rates $r_1 = 2$ and $r_2 = r_3 = 1$ are in the system. $w_1 = 1/2$, $w_2 = 1/4$, and $w_3 = 1/4$. Thus, f_1 is in F_1, and f_2 and f_3 are in F_2. Let L be the maximum packet size. The quantum for each of the three flows is L. All packets in f_1 are of size L, all packets in f_2 are of size $0.99L$ and all packets in f_3 are of size $0.01L$. Flows f_1 and f_2 are always backlogged. Flow f_3 is not always backlogged, its packets arrive in such a way that exactly one packet arrives before a new frame is to be formed. Thus, each F_2 frame contains one packet from f_3. The example is depicted in Fig. 2. As shown in the figure, each F_1 frame contains exactly one packet from f_1. For F_2, the lookahead operation always moves part of the f_2 packet in the next DRR round into the current frame, and thus, the frame boundaries are not aligned with packet boundaries. Note that frames from different classes are not aligned. We will shows how worst-case and proportional fairness for flows f_1 and f_2 is achieved.

The weight for F_1, which is computed using line (48) in Fig. 1, is always $1/2$. For F_2, the lookahead operation ensures that the size of f_2 data in a frame is L, and thus, the size of each F_2 frame is $L + 0.01L = 1.01L$. The weight of F_2 is computed using line (48) in Fig. 1. $weight = (w_2 + w_3) * \frac{framesize}{quantum_2 + quantum_3} = (\frac{1}{4} + \frac{1}{4}) * \frac{1.01L}{2L} = \frac{1.01}{4}$. Hence, F_1 (and thus f_1) is allocated a bandwidth of $4 * \frac{\frac{1}{2}}{\frac{1}{2} + \frac{1.01}{4}} = \frac{8}{3.01} > 2$. F_2 is allocated a bandwidth of $4 * \frac{\frac{1.01}{4}}{\frac{1}{2} + \frac{1.01}{4}}$. For each f_2 frame of size $1.01L$, L belongs to f_2. Thus, the rate allocated to f_2 is $4 * \frac{\frac{1.01}{4}}{\frac{1}{2} + \frac{1.01}{4}} * \frac{L}{1.01L} = \frac{4}{3.01} > 1$. The rates allocated to f_1 and f_2 are larger than their guaranteed rates and the worst-case fairness is honored. Furthermore, the ratio of the rates allocated to f_1 and f_2 is equal to $\frac{\frac{8}{3.01}}{\frac{4}{3.01}} = 2$, which is equal to the ratio of their weights. Thus, the proportional fairness is also honored. In the next section, we will formally prove that FRR has worst-case and proportional fairness.

V. Properties of FRR

This session analyzes fairness and delay properties of FRR. We will prove that the three statements in Lemma 2 hold for FRR with an arbitrary weight distribution.

Theorem 5 (single packet delay bound): Let packet p arrives at the head of flow $f_i \in F_k$ at time t. Using FRR, there exists a constant c_1 such that p will depart before $t + c_1 * \frac{L_M}{r_i}$.

Proof: If class F_k is idle under GPS at time t, a new frame that includes packet p will be formed at time t. From Lemma 6, under GPS, the frame will be serviced at most at time $t + C^k \frac{L_M}{R} \leq t + C\frac{L_M}{r_i}$. Hence, from Theorem 2, the frame will be serviced under DW^2F^2Q before $t + C\frac{L_M}{r_i} + (n-1)\frac{L_M}{R} \leq t + (C + n - 1)\frac{L_M}{r_i}$, where n is the number of classes in the system. Thus, there exists $c_1 = C + n - 1$ such that packet departs before $t + c_1 * \frac{L_M}{r_i}$.

If class F_k is busy under GPS at time t, packet p will be included in the frame that is computed the next time the frame computation algorithm is invoked, which is at the end of the current frame under consideration by GPS. From Lemma 6, $F_{GPS}^p \leq t + 2 * \frac{L_M C^k}{R} \leq t + \frac{2CL_M}{r_i}$. From Theorem 2, the frame will be serviced under DW^2F^2Q before $t + 2C\frac{L_M}{r_i} + (n -$

$1)\frac{L_M}{R} \leq t+(2C+n-1)\frac{L_M}{r_i}$. Thus, there exists $c_1 = 2C+n-1$ such that packet p departs before $t + c_1 * \frac{L_M}{r_i}$. □

Theorem 5 shows that like the stratified round robin scheme [11], FRR also has a worst-case single packet delay bound that is only related to requested rate of the flow and is independent of the number of flows in the system. Next, we will consider the worst-case fairness property of FRR.

Theorem 6 (worst-case fairness): FRR has a constant normalized worst-case fairness index.

Proof: Let a packet belonging to flow $f_i \in F_k$ arrive at time t, creating a total backlog of q_i bytes in f_i's queue. Let packet p_1 be the first packet in the backlog. From the proof of Theorem 5, we have $F_{GPS}^{p_1} \leq t + 2C\frac{L_M}{r_i}$. After the first packet is serviced under GPS, from Lemma 4, at most $\lceil \frac{q_i}{quantum_i} \rceil + 4 \leq \frac{q_i}{quantum_i} + 5$ frames will be needed to drain the queue. From Lemma 6, under GPS, servicing the $\frac{q_i}{quantum_i} + 5$ frames will take at most

$$(\frac{q_i}{quantum_i} + 5) * C^k \frac{L_M}{R} = \frac{q_i}{C^k w_i L_M}\frac{C^k L_M}{R} + 5C^k \frac{L_M}{R} \leq \frac{q_i}{r_i} + 5C\frac{L_M}{r_i}$$

Thus, under GPS, the queue will be drained before $t + \frac{q_i}{r_i} + 2C\frac{L_M}{r_i} + 5C\frac{L_M}{r_i}$. From Theorem 2, under DW^2F^2Q, the queue will be drained before $t+\frac{q_i}{r_i}+7C\frac{L_M}{r_i}+(n-1)\frac{L_M}{R}$. Thus, there exists a constant $d = 7C+n-1$ such that the queue will be drained before $t+\frac{q_i}{r_i}+d\frac{L_M}{r_i}$ and the normalized worst-case fair index for FRR is $\max_i\{\frac{r_i * d\frac{L_M}{r_i}}{R}\} = d\frac{L_M}{R}$, which is a constant. □

To the best of our knowledge, FRR is the first O(1) complexity scheduling scheme that has a constant worst-case fairness index. Next we will consider FRR's proportional fairness.

Lemma 12: Assuming that $f_i \in F_k$ and $f_j \in F_m$ are continuously backlogged during $[t_1, t_2)$, $k \geq m$. Assume that the inter-class scheduler is GPS and the intra-class scheduler is $LDRRWA$. Let $S_i(t_1, t_2)$ be the services given to flow f_i during $[t_1, t_2)$ and $S_j(t_1, t_2)$ be the services given to flow f_j during $[t_1, t_2)$. There exists two constants c_1 and c_2 such that

$$|\frac{S_i(t_1, t_2)}{r_i} - \frac{S_j(t_1, t_2)}{r_j}| \leq \frac{c_1 * L_M}{r_i} + \frac{c_2 * L_M}{r_j}.$$

Proof: Let X_k and X_m be the smallest numbers of F_k and F_m frames that completely enclose $[t_1, t_2)$. Since f_i and f_j are continuously backlogged during the $[t_1, t_2)$ period, from Lemma 4, the services given to f_i and f_j during this period satisfy:

$$(X_k - 4)quantum_i \leq S_i(t_1, t_2) \leq (X_k + 2)quantum_i$$

and

$$(X_m - 4)quantum_j \leq S_j(t_1, t_2) \leq (X_m + 2)quantum_j.$$

The conclusion follows by manipulating these in-equations and applying Lemma 11, which gives the relation between X_k and X_m,

$$(X_k - 1)C^{k-m} \leq X_m \leq X_k C^{k-m} + 1.$$

In the following, we will derive the bound for $\frac{S_i(t_1,t_2)}{r_i} - \frac{S_j(t_1,t_2)}{r_j}$. We will have the term CON to represent the general term $c_1\frac{L_M}{r_i} + c_2\frac{L_M}{r_j}$, where c_1 and c_2 are constants.

$$\begin{aligned}
&\frac{S_i(t_1,t_2)}{r_i} - \frac{S_j(t_1,t_2)}{r_j} \\
&\leq \frac{(X_k+2)quantum_i}{r_i} - \frac{(X_m-4)quantum_j}{r_j} \\
&\leq \frac{quantum_i X_k}{r_i} - \frac{quantum_j X_m}{r_j} + CON \\
&\leq \frac{quantum_i X_k}{r_i} - \frac{quantum_j (X_k-1)C^{k-m}}{r_j} + CON \\
&= \frac{quantum_i X_k}{r_i} - \frac{quantum_j (X_k)C^{k-m}}{r_j} \\
&\quad + \frac{quantum_j C^{k-m}}{r_j} + CON
\end{aligned}$$

We have $\frac{quantum_j C^{k-m}}{r_j} = \frac{C^m w_j L_M C^{k-m}}{w_j R} \leq \frac{C*L_M}{r_i}$ and

$$\begin{aligned}
&\frac{quantum_i X_k}{r_i} - \frac{quantum_j (X_k)C^{k-m}}{r_j} \\
&= \frac{C^k w_i L_M X_k}{w_i R} - \frac{C^m w_j L_M X_k C^{k-m}}{w_j R} = 0
\end{aligned}$$

Thus, $\frac{S_i(t_1,t_2)}{r_i} - \frac{S_j(t_1,t_2)}{r_j} \leq CON$. The bound for $\frac{S_j(t_1,t_2)}{r_j} - \frac{S_i(t_1,t_2)}{r_i}$ can be derived in a similar fashion. □

Lemma 12 shows that if GPS is used as the inter-class scheduling algorithm, the scheduling algorithm provides proportional fairness. Since DW^2F^2Q approximates GPS, we will show in the next theorem that FRR, which uses DW^2F^2Q as the inter-class scheduling algorithm, also supports proportional fairness.

Theorem 7 (proportional fairness): In any time period $[t_1, t_2)$ during which flows $f_i \in F_k$ and $f_j \in F_m$ are continuously backlogged in FRR. There exists two constants c_1 and c_2 such that

$$|\frac{S_{i,FRR}(t_1, t_2)}{r_i} - \frac{S_{j,FRR}(t_1, t_2)}{r_j}| \leq \frac{c_1 * L_M}{r_i} + \frac{c_2 * L_M}{r_j}.$$

Proof: There are two cases. The first case is when flows f_i and f_j are in the same class, that is, $k = m$. The second case is when flows f_i and f_j are not in the same class, that is, $k \neq m$. The proof of the first case is similar to the proof of the statement 3 in Lemma 2. Here, we will focus on the second case. Let us assume that $k > m$.

Let packets p_k^1, p_k^2, ..., p_k^a be the sequence of class F_k packets sent under FRR during $[t_1, t_2)$. Let packets p_m^1, p_m^2, ..., p_m^b be the sequence of class F_m packets sent under FRR during $[t_1, t_2)$. Since flows f_i and f_j are continuously backlogged during $[t_1, t_2)$, there exists a packet p_k^0 that departed before p_k^1 and p_k^{a+1} that will depart after p_k^a. Under the simulated GPS, there is no idle time between packet p_k^0 and packet p_k^1 and between packet p_k^a and packet p_k^{a+1}. Packets p_m^0 and p_m^{b+1} are defined similarly.

Consider the progress of these packets under the simulated GPS. Let B_{GPS}^p denote the beginning time of packet p under GPS and F_{GPS}^p denote the finishing time of packet p under GPS. There are four cases: (1) $B_{GPS}^{p_k^1} \geq B_{GPS}^{p_m^1}$ and $F_{GPS}^{p_k^a} < F_{GPS}^{p_m^b}$, (2) $B_{GPS}^{p_k^1} \geq B_{GPS}^{p_m^1}$ and $F_{GPS}^{p_k^a} \geq F_{GPS}^{p_m^b}$, (3) $B_{GPS}^{p_k^1} < B_{GPS}^{p_m^1}$ and $F_{GPS}^{p_k^a} < F_{GPS}^{p_m^b}$, and (4) $B_{GPS}^{p_k^1} < B_{GPS}^{p_m^1}$ and $F_{GPS}^{p_k^a} \geq F_{GPS}^{p_m^b}$.

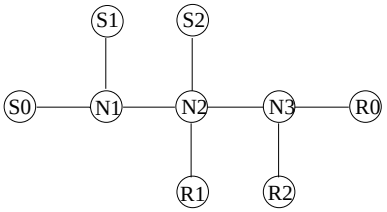

Fig. 3. Simulated network.

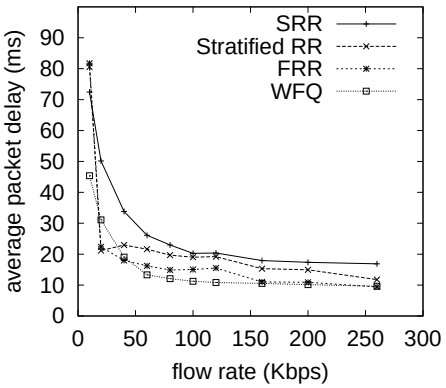

Fig. 4. Average end-to-end delay.

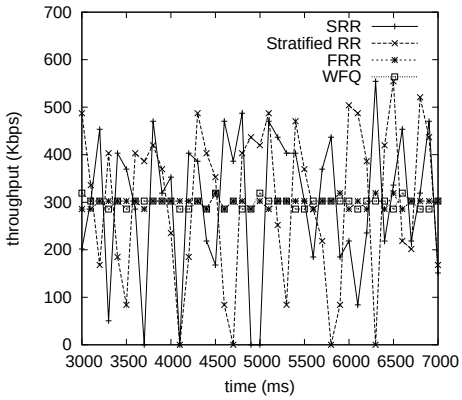

Fig. 5. Short-term throughput

In the next, we will prove case (1). Other three cases can be proven in a similar fashion. Consider case (1) when

$$B_{GPS}^{p_k^1} \geq B_{GPS}^{p_m^1} \text{ and } F_{GPS}^{p_k^a} < F_{GPS}^{p_m^b}$$

Let $tt_0 = B_{GPS}^{p_m^1}$, $tt_1 = B_{GPS}^{p_k^1}$, $tt_2 = F_{GPS}^{p_k^a}$, and $tt_3 = F_{GPS}^{p_m^b}$. We have $tt_0 \leq tt_1 \leq tt_2 \leq tt_3$. Let $S_{i,GPS}(t_1, t_2)$ be the services that flow f_i received during time $[t_1, t_2)$ in the simulated GPS. We have

$$S_{i,FRR}(t_1, t_2) = S_{i,GPS}(tt_1, tt_2)$$

$$S_{j,FRR}(t_1, t_2) = S_{j,GPS}(tt_0, tt_1) + S_{j,GPS}(tt_1, tt_2) + S_{j,GPS}(tt_2, tt_3)$$

In the simulated GPS system, flows f_i and f_j are continuously backlogged during $[tt_1, tt_2)$. From Lemma 12, there exist two constants cc_1 and cc_2 such that

$$|\frac{S_{i,GPS}(tt_1, tt_2)}{r_i} - \frac{S_{j,GPS}(tt_1, tt_2)}{r_j}| \leq \frac{cc_1 * L_M}{r_i} + \frac{cc_2 * L_M}{r_j}.$$

Thus,

$$\begin{aligned}
&|\frac{S_{i,FRR}(t_1,t_2)}{r_i} - \frac{S_{j,FRR}(t_1,t_2)}{r_j}| \\
&\leq |\frac{S_{i,GPS}(tt_1,tt_2)}{r_i} - \frac{S_{j,GPS}(tt_1,tt_2)}{r_j}| + \frac{S_{j,GPS}(tt_0,tt_1)}{r_j} + \frac{S_{j,GPS}(tt_2,tt_3)}{r_j} \\
&\leq \frac{cc_1*L_M}{r_i} + \frac{cc_2*L_M}{r_j} + \frac{S_{j,GPS}(tt_2,tt_3)}{r_j} + \frac{S_{j,GPS}(tt_0,tt_1)}{r_j}
\end{aligned}$$

Next, we will consider the two terms $\frac{S_{j,GPS}(tt_0,tt_1)}{r_j}$ and $\frac{S_{j,GPS}(tt_2,tt_3)}{r_j}$. First, consider class F_m packets serviced during $[tt_0, tt_1)$. Since all these packets are serviced after packet p_k^0 under FRR (DW^2F^2Q as the inter-class scheduler), from Lemma 3, at most one of the packets can have a GPS finishing time before $F_{GPS}^{p_k^0} = B_{GPS}^{p_k^1} = tt_1$. That is, there can be at most one class F_m packet finishing during $[tt_0, tt_1)$. Thus, in the simulated GPS, at most two class F_m packets can be serviced during $[tt_0, tt_1)$ and

$$\frac{S_{j,GPS}(tt_0, tt_1)}{r_j} \leq \frac{2L_M}{r_j}.$$

Now, consider class F_m packets serviced during $[tt_2, tt_3)$. Since all these packets are serviced under FRR before packet p_k^{a+1}, at most one of the packets can have a GPS finishing time after $F_{GPS}^{p_k^{a+1}}$. From Lemma 7, the duration of packet p_k^{a+1} is less than $\frac{C^m L_M}{R}$ in the simulated GPS, which is less than one frame whose size is larger than L_M. Let X be the number of frames for class F_m during this period when p_k^{a+1} is in progress under GPS. Since f_j is continuously backlogged during this period of time, from Lemma 11, $X \leq C^{k-m} * 1 + 1$. Thus, from Lemma 4, during the period that packet p_k^{a+1} is in progress under GPS, the amount of services given to flow f_j is at most $(C^{k-m} + 1 + 2)quantum_j$.

$$\begin{aligned}
\frac{S_{j,GPS}(tt_2,tt_3)}{r_j} &\leq \frac{(C^{k-m}+1+2)quantum_j + L_M}{r_j} \\
&= \frac{(C^{k-m}+3)w_j C^m L_M + L_M}{w_j R} \leq \frac{L_M C^k}{R} + \frac{3L_M C^m}{R} + \frac{L_M}{r_j} \\
&\leq C\frac{L_M}{r_i} + (3C+1)\frac{L_M}{r_j}
\end{aligned}$$

Thus, there exists two constants $c_1 = cc_1 + C$ and $c_2 = cc_2 + 2 + 3C + 1$ such that

$$|\frac{S_{i,FRR}(t_1, t_2)}{r_i} - \frac{S_{j,FRR}(t_1, t_2)}{r_j}| \leq \frac{c_1 * L_M}{r_i} + \frac{c_2 * L_M}{r_j}.$$

□

VI. Simulation Experiments

In this section, we report the results of our simulation experiments. These experiments are designed to investigate FRR properties in practical situations and to compare FRR with other scheduling disciplines, including Weighted Fair Queueing (WFQ) and two recently proposed deficit round robin (DRR) based schemes, Smoothed Round Robin (SRR) [6] and STratified Round Robin ($STRR$) [11]. All experiments are performed using *ns-2* [9], to which we added WFQ, $STRR$, and FRR queuing classes. While we carried out extensive simulations, we will only report the results of two representative experiments, one for end-to-end delay and the other one for short-term throughput. Fig. 3 shows the network topology used in the experiments. All the links have a bandwidth of $2\,Mbps$ and a propagation delay of $1\,ms$.

The first experiment shows the end-to-end delay for flows with different rates. In this experiment, there are 10 CBR flows from $S0$ to $R0$ with average rates of $10Kbps$, $20Kbps$, $40Kbps$, $60Kbps$, $80Kbps$, $100Kbps$, $120Kbps$, $160Kbps$, $200Kbps$, and $260Kbps$. The packet delay of these ten CBR flows are measured. In addition to the ten observed flows, the background traffic in the system is as follows. There are five exponential on/off flows from $S1$ to $R1$ with rates $60Kbps$, $80Kbps$, $100Kbps$, $120Kbps$, and $160Kbps$. The on-time and the off-time are 0.5 second. There are five Pareto on/off flows from $S2$ to $R2$ with rates $60Kbps$, $80Kbps$, $100Kbps$,

$120Kbps$, and $160Kbps$. The on-time and the off-time are 0.5 second. The shape parameter of the Pareto flows is 1.5. Two $7.8Kbps$ FTP flows with infinite traffic are also in the system, one from $S1$ to $R1$ and the other one from $S2$ to $R2$. CBR flows have a fixed packet size of 210 bytes, and all other flows have a fixed packet size uniformly chosen between 128 bytes and 1024 bytes.

Fig. 4 shows the average end-to-end delays for the ten CBR flows. From the figure we can see that FRR achieves average end-to-end delays that are close to the ones that can be provided by WFQ, especially for flows with large rates (above $150\,Kbps$ in the experiment). In FRR, the timestamp based inter-class scheduling mechanism is added on top of DRR so that flows with small rates do not significantly affect flows with large rates. Thus, in a way, FRR gives preference to flows with larger weights in comparison to other DRR bases schemes. In this experiment, FRR results in smaller average end-to-end delays than SRR and $STRR$ when the flow rate is larger than $10Kbps$, while having a larger packet delay for the $10Kbps$ flow.

The second experiment is designed to demonstrate that FRR has a better short-term throughput property than existing DRR based schemes. As discussed earlier, in all existing DRR based schemes, the short-term throughput of a flow with a large rate can be significantly affected by flows with small rates. In this experiment, we observe one $300Kbps\,CBR$ flow and one $600Kbps$ flow from $S0$ to $R0$. In addition, we have 50 $10Kbps\ CBR$ flows from $S0$ and $R0$. The background flows are the same as the previous experiment.

Fig. 5 shows the short-term thoughput of the 300 $Kbps$ flow with different scheduling schemes. The results for the 600 $Kbps$ flow show a similar trend. Each point in the figure represents the throughput in an interval of 100ms. As can be seen from the figure, the short term throughputs for both SRR and $STRR$ exhibit heavy fluctuations. The flow may significantly under–perform or over–perform for a period of up to 400 ms. For example, SRR significantly under-performs between 5800ms and 6200ms and $STRR$ significantly over-performs between 4700ms and 5100ms. On the other hand, WFQ and FRR yield much better short term thoughputs: within each interval of 100ms, the throughputs are always close to the ideal rate. This experiment demonstrates that FRR has a better short-term throughput property than SRR and $STRR$ and is immune to impacts of many low-speed flows on the high-speed flows.

VII. Conclusion

In this paper, we have described a new scheduling algorithm, Fair Round Robin (FRR). We demonstrate that FRR has the desired properties of an ideal packet scheduler: an O(1) per packet processing complexity, a strict rate-proportional delay bound, and proportional and worst-case fairness. In particular, FRR has much better short term fairness than other recently proposed DRR based schemes including smoothed round robin and stratified round robin. The constant factor in the per packet processing complexity of FRR is fairly large. We are currently investigating techniques to reduce the constant factor.

References

[1] J. Bennett and H. Zhang, "Hierarchical Packet Fair Queueing Algorithms," in *ACM SIGCOMM'96* (1996).

[2] J. Bennett and H. Zhang, "WF^2Q: Worst Case Fair Weighted Fair Queuing", in *IEEE INFOCOM'96* (1996), pages 120-128.

[3] S.Cheung and C. Pencea, "BSFQ: Bin Sort Fair Queuing," in *IEEE INFOCOM'02* (2002).

[4] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queuing Algorithm," in *ACM SIGCOMM'89* (1989).

[5] S. Golestani, "A Self-clocked Fair Queueing Scheme for Broadband Applications", in *IEEE INFOCOM'94*, 1994.

[6] C. Guo, "SRR, an O(1) Time Complexity Packet Scheduler for Flows in Multi-Service Packet Networks," in *ACM SIGCOMM'01* (2001).

[7] L Lenzini, E. Mingozzi, and G. Stea, "Aliquem: a Novel DRR Implementation to Achieve Better Latency and Fairness at O(1) Complexity," in *IWQoS'02* (2002).

[8] L. Massouli and J. Roberts, "Bandwidth Sharing: Objectives and Algorithms," *IEEE/ACM Trans. on Networking*, Vol. 10, No. 3, pages 320-328, June 2002.

[9] "The Network Simulator - ns-2," available from http://www.isi.edu/nsnam/ns.

[10] A. Parekh and R. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: the Single Node Case," *IEEE/ACM Transaction on Networking*, Vol. 1, No. 3, pages 344-357, June 1993.

[11] S. Ramabhadran and J. Pasquale, "Stratified Round Robin: A Low Complexity Packet Scheduler with Bandwidth Fairness and Bounded Delay," in *ACM SIGCOMM'03* (2003), pages 239-249.

[12] J. Rexford and A. Greenberg and F. Bonomi, "Hardware-Efficient Fair Queueing Architectures for High-Speed Networks," *IEEE INFOCOM'96*, 1996.

[13] J. L. Rexford, F. Bonomi, A. Greenberg, A. Wong, "A Scalable Architecture for Fair Leaky-Bucket Shaping," in *INFOCOM'97* (1997), pages 1056–1064.

[14] D. Stiliadis and A. Varma, "Design and Analysis of Frame-based Fair Queueing: A New Traffic Scheduling Algorithm for Packet-Switched Networks," in *ACM SIGMETRICS'96*, (1996).

[15] M. Shreedhar and G. Varghese, "Efficient Fair Queuing using Deficit Round Robin," in *ACM SIGCOMM'95* (1995), pages 231-242.

[16] S. Suri, G. Varghese, and G Chandranmenon, "Leap Forward Virtual Clock: An $O(loglog\ N)$ Queuing Scheme with Guaranteed Delays and Throughput Fairness," in *IEEE INFOCOM'97* (1997).

[17] J. Xu and R. J. Lipton, "On Fundamental Tradeoffs between Delay Bounds and Computational Complexity in Packet Scheduling Algorithms," in *ACM SIGCOMM'02* (2002).

[18] X. Yuan and Z. Duan, "FRR: a Proportional and Worst-Case Fair Round Robin Scheduler," Technical Report TR-040201, Department of Computer Science, Florida State Univerity, Feb. 2004. http://www.cs.fsu.edu/∼xyuan/frrtr.pdf

[19] L. Zhang, "Virtual Clock: A New Traffic Control Scheme for Packet Switching Networks", in *ACM SIGCOMM'90* (1990).