Group Round Robin: Improving the Fairness and Complexity of Packet Scheduling

Bogdan Caprita, Jason Nieh, and Wong Chun Chan
Dept. of Computer Science, Columbia University
New York, NY, USA
bc2008@columbia.edu, nieh@cs.columbia.edu, wc164@cs.columbia.edu

ABSTRACT

We present Group Round-Robin (GRR) scheduling, a hybrid fair packet scheduling framework based on a grouping strategy that narrows down the traditional trade-off between fairness and computational complexity. GRR combines its grouping strategy with a specialized round-robin scheduling algorithm that utilizes the properties of GRR groups to schedule flows within groups in a manner that provides $O(1)$ bounds on fairness with only $O(1)$ time complexity. Under the practical assumption that GRR employs a small constant number of groups, we apply GRR to popular fair queuing scheduling algorithms and show how GRR can be used to achieve constant bounds on fairness and time complexity for these algorithms. We also present and prove new results on the fairness bounds for several of these fair queuing algorithms using a consistent fairness measure. We analyze the behavior of GRR and present experimental results that demonstrate how GRR can be combined with existing scheduling algorithms to provide much lower scheduling overhead and more than an order of magnitude better scheduling accuracy in practice than scheduling algorithms without GRR.

Categories and Subject Descriptors: F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems - *Sequencing and scheduling*

General Terms: Algorithms

Keywords: Stochastic Processes/Queuing Theory, Quality of Service, Scheduling, Fair Queuing

1. INTRODUCTION

Packet scheduling is an important mechanism for meeting the quality-of-service needs of competing flows in packet-switched data networks. An important class of packet schedulers are those that treat flows in a proportionally fair manner. Given a set of flows with associated weights, these schedulers try to allocate network link capacity to each flow in proportion to its respective weight. The goal of these schedulers is to provide the highest degree of fairness in allocating link capacity with the lowest time complexity of scheduler execution.

ANCS'05, October 26–28, 2005, Princeton, New Jersey, USA.

Many packet scheduling algorithms have been proposed with different trade-offs in fairness and time complexity. The fairest one is Generalized Processor Sharing (GPS) [13], which is an idealized fluid model that services flows continuously and simultaneously, but cannot be employed in practice since packets must be transmitted as a unit. Algorithms such as Weighted Fair Queuing (WFQ) [9] use a notion of virtual time to emulate GPS and approximate its behavior. However, WFQ can allow the service that a flow receives to deviate from service under GPS in the worst case by $O(N)$, where N is the number of flows being scheduled. Since GPS provides ideal max-min fairness, deviations in service from GPS are considered service error. Furthermore, WFQ requires at least $O(\log N)$ time complexity. Other variants of WFQ such as Virtual-clock [22], SFQ [12], SCFQ [10], SPFQ [17], and Time-shift FQ [8] have also been proposed. However, these algorithms share the same fairness and time complexity bounds as WFQ, allowing service errors of $O(N)$ in the worst case and requiring $O(\log N)$ time complexity. Worst-Case Weighted Fair Queuing (WF^{2}Q) [1] and its variants introduce eligible virtual times to provide stronger fairness guarantees than previous fair queuing approaches, limiting deviations in service error to $O(1)$. However, these algorithms still require at least $O(\log N)$ time complexity.

Because $O(\log N)$ time complexity scheduling is not enough high-speed links [7], there is great interest in developing lower complexity packet schedulers that still provide good fairness guarantees. Round-robin packet schedulers such as Deficit Round-Robin (DRR) [16] have been developed, requiring only $O(1)$ time complexity by servicing each flow for a continuous amount of time proportional to its weight. However, these round-robin schedulers deviate much more from GPS service than virtual time fair queuing approaches, with worst case service errors of $O(\phi_{\max})$, where $\phi_{\max}$ is the maximum flow weight. More recent schedulers like Smooth Round-Robin (SRR) [7] provide in practice better fairness than previous round-robin approaches while retaining $O(1)$ time complexity. However, we show that the service error of SRR is still worse than $O(N)$. Another recent approach is Stratified Round-Robin [15], which still has $O(N)$ service error and does not have the desired $O(1)$ time complexity.

We present Group Round-Robin (GRR) scheduling, a flexible scheduling framework that can be used with existing packet scheduling algorithms to improve the overall fairness and reduce the time complexity of scheduling competing flows. GRR is based on a group strategy that can be used to allow existing schedulers to consider groups of flows together instead of individually. The use of a binary grouping strategy was first introduced in [5] and applied to link scheduling in [3]. GRR generalizes this idea along with providing detailed analysis and experimental results. Fair Round-Robin (FRR) [21] exemplifies an application of GRR to the WF^2Q

([1]) scheduler. Similar groupings were employed to improve fairness and running time by Stratified Round-Robin (StRR) [15] and, in process scheduling, by Group Ratio Round-Robin (GR^3) [4].

Reducing the number of competing entities that existing schedulers must consider can improve both fairness and reduce time complexity when such properties depend on the number of entities scheduled. GRR's grouping strategy requires only simple queues for groups of flows, allowing it to be easily implemented in an efficient manner. GRR combines its grouping strategy with a specialized round-robin scheduling algorithm that utilizes the properties of GRR groups to schedule flows within groups in a manner that limits service error to $O(1)$ with only $O(1)$ time complexity. We prove analytically and demonstrate experimentally how GRR can be applied to existing scheduling algorithms to improve their fairness and time complexity bounds. The algorithms described include WFQ [9], SCFQ [10], SFQ [12], Hierarchical stride [19], WF^{2}Q [1], and SRR [7]. We also present new results on the fairness bounds for several of these algorithms using a consistent fairness measure across all algorithms, where such results are missing. Under the practical assumption that GRR employs a small constant number of groups, we show how GRR can be used to achieve constant bounds on fairness and time complexity for any of these algorithms. The resulting GRR-based scheduling algorithms provide better fairness bounds than any other constant time scheduling algorithms.

This paper presents the design and analysis of GRR. Section 2 provides some background and discusses more precisely the notion of fairness. Section 3 describes the GRR scheduling algorithm. Section 4 analyzes the fairness and time complexity of GRR. Section 5 describes how GRR can be applied to several existing scheduling algorithms to improve their fairness and time complexity bounds. Section 6 discusses the delay bounds for GRR. Section 7 presents some performance results from simulation studies to compare the fairness of existing scheduling algorithms with the same algorithms augmented with GRR. Section 8 considers related work. Finally, we present some concluding remarks.

2. BACKGROUND

We first define the general terminology and state the assumptions we will use throughout the paper. Table 1 is a list of the general terminology we use. We consider work-conserving servers that use their maximum available bandwidth during any busy period, defined as a time interval during which there is at least one backlogged flow at any time. A flow is backlogged at time t if it has bits left that are waiting to be sent, $L(Q_i(t))>0$. Since we are mainly concerned with busy periods, for any function $\Gamma(t_1, t_2)$, we will use the shorthand notation $\Gamma(t)$ to mean $\Gamma(t_0, t)$ for a busy period that started at t_0. For example, $W(t)=\int_{t_0}^{t} Rdt$. Each flow S_i has a share ϕ_i assigned to it, which determines the proportional allocation of bandwidth in the system. Although similar concepts, we distinguish between ϕ_i and R_i, the rate guaranteed to flow S_i, to allow for greater flexibility in defining the scheduling policy and for a tighter analysis of fairness even in the case when the bandwidth R is not fully utilized. If flows are to receive guaranteed minimum rates, this amounts to having a cap on the total share in the system: $\phi(t) \leq \phi^{MAX}$. Then $R_i = \frac{\phi_i}{\phi^{MAX}} R$. On the other hand, some schedulers might choose not to implement admission control and accept any number of new flows at the expense of degrading bandwidth and delay allocations. Our analysis captures the correct fairness bounds under these conditions as well.

To compare the performance of schedulers, we need to use a consistent measure of fairness. For this purpose, we describe two measures of fairness prevalent in the literature. We start with the relative fairness of two flows, $F_{i,j}$, since this is widely used as an indicator of fairness ([6], [7], [10], [11], [15]). Intuitively, $F_{i,j}$ measures the discrepancy in normalized service received by two backlogged flows i and j that ideally are supposed to be accurately serviced in proportion to their weight. A theorem given without proof in [10] states that if a packet-based algorithm guarantees $F_{i,j}(t_1, t_2) \leq M_{i,j}$ for any interval (t_1, t_2) when S_i and S_j are backlogged, then $M_{i,j}$ must be at least $\frac{1}{2}(\frac{L_{\max}}{\phi_i} + \frac{L_{\max}}{\phi_j})$. $M_{i,j}$ is a theoretical bound on the relative fairness which usually depends on the weights of the flows and the packet size. We found a tighter result that shows relative fairness bounds of $\frac{L_{\max}}{\phi_i} + \frac{L_{\max}}{\phi_j}$ are the best that can be achieved by any discrete packet scheduling algorithm. The proof can be found in the Appendix 9. Such lower relative fairness bounds are used by first proving that $M_{i,j}$ is less than $\kappa(\frac{L_{\max}}{\phi_i} + \frac{L_{\max}}{\phi_j})$ for a given scheduling algorithm and then making the claim that a small constant value of κ implies good fairness for this algorithm.

Table 1: General Terminology

S_i	Flow i.
p_i^k	The k^{th} packet to have arrived for S_i.
$B(t)$	The set of backlogged flows at time t.
N	The number of backlogged flows.
$a(p)$	The arrival time of packet p.
$d(p)$	The departure time of packet p.
$L_{i,\max}$	The maximum size of a packet for S_i.
$L_{\max}$	The maximum size of a packet.
Q_i(t)	The queue of S_i at time t.
$L(Q_i(t))$	The number of bits on $Q_i(t)$.
ϕ_i	The weight assigned to S_i.
$\phi(t)$	The sum of the weights of all flows backlogged at time t: $\sum_{S_j \in B(t)} \phi_j$.
R	The output link bandwidth of the server.
R_i	The guaranteed rate for flow S_i.
$W_i(t_1, t_2)$	The amount of traffic served from S_i.
$w_i(t_1, t_2)$	The normalized work received by flow S_i: $\frac{W_i(t_1,t_2)}{\phi_i}$.
$W(t_1, t_2)$	The total traffic served by the server.
$w(t_1, t_2)$	The total normalized work: $\frac{W(t_1,t_2)}{\phi}$, where $B(t)$ is fixed during (t_1, t_2).
$F_{i,j}(t_1, t_2)$	The relative fairness of flows S_i and S_j: $\lvert w_i(t_1, t_2) - w_j(t_1, t_2)\rvert$.
$F_i(t_1, t_2)$	The absolute fairness: $\lvert w_i(t_1, t_2) - w(t_1, t_2)\rvert$.
$E_i(t_1, t_2)$	The absolute service error: $\phi_i F_i(t_1, t_2) = \lvert W_i(t_1, t_2) - \frac{\phi_i}{\phi} W(t_1, t_2)\rvert$.

While relative fairness can be a useful measure, it is a weaker measure compared to the absolute service error $E_i(t_1, t_2)$, which relates the work of a flow to the allocation the flow should have received given the work done by the scheduler. This measure is inspired by the GPS-relative error introduced in [14] and is a compelling indicator of scheduling inaccuracy ([1], [15], [19]). However, there is one caveat with defining such a measure of fairness: while most schedulers take their own set of backlogged flows $B(t)$ into account when assigning service, some time-stamp schedulers such as WFQ and WF^{2}Q have as reference $B^{GPS}(t)$, the set of backlogged flows under the GPS server they are simulating. For the latter, we need to use $B^{GPS}(t)$ in our definition of w. Otherwise, we would discover that F_i and E_i may grow infinitely large with time. With this observation in mind, the service error does in fact reduce to a measure of service error versus GPS assuming that

all flows are continuously backlogged. This is an assumption we will make during fairness analysis for the intervals we consider.

To see now that absolute service error is a stronger measure than relative fairness, suppose we have service error $E_i \leq \kappa L_{i,\max}$, then the relative fairness is bounded as follows:

$$\begin{aligned} F_{i,j} &= |w_i - w_j| \leq |w_i - w| + |w_j - w| \\ &= \frac{E_i}{\phi_i} + \frac{E_j}{\phi_j} \leq \kappa(\frac{L_{i,\max}}{\phi_i} + \frac{L_{j,\max}}{\phi_j}) \end{aligned} \quad (1)$$

So an $O(1)$ bound on error (κ constant) always implies an $O(1)$ bound on relative fairness.

To see that relative fairness is a weaker fairness measure, we also show that an $O(1)$ bound on the relative fairness of an algorithm only translates to an $O(N)$ bound on the absolute service error. The service error for a flow S_i is

$$\begin{aligned} E_i(t_1,t_2) &= |W_i(t_1,t_2) - \frac{\phi_i W(t_1,t_2)}{\phi}| \\ &= |\frac{\phi_i w_i(t_1,t_2)\sum_j \phi_j}{\phi} - \frac{\phi_i \sum_j \phi_j w_j(t_1,t_2)}{\phi}| \\ &\leq \frac{\phi_i \sum_j \phi_j |w_i(t_1,t_2) - w_j(t_1,t_2)|}{\phi} \\ &= \frac{\phi_i \sum_{j\neq i} \phi_j F_{i,j}(t_1,t_2)}{\phi}. \end{aligned}$$

For an algorithm guaranteeing $F_{i,j}(t_1,t_2) \leq \kappa(\frac{L_{i,\max}}{\phi_i} + \frac{L_{j,\max}}{\phi_j})$,

$$\begin{aligned} E_i(t_1,t_2) &\leq \tfrac{1}{\phi}(\phi_i \textstyle\sum_{j\neq i} \phi_j \kappa(\frac{L_{i,\max}}{\phi_i} + \frac{L_{j,\max}}{\phi_j})) \\ &\leq \kappa L_{\max}(1 + (N-2)\tfrac{\phi_i}{\phi}) \\ &\leq \kappa(N-1)L_{\max} \end{aligned} \quad (2)$$

Note that ϕ_i can get arbitrarily close to ϕ, and thus E_i can grow linearly in N.

To show that the $O(1)$ bounds on relative fairness, $\kappa(\frac{L_{i,\max}}{\phi_i} + \frac{L_{j,\max}}{\phi_j})$, cannot imply a service error bound of less than $O(N)$, consider an example of N flows, the first having weight $N-1$, and the rest having weights equal to 1. For instance, the SCFQ algorithm, for which the $O(1)$ relative fairness bound holds [10], has service error of $\frac{N-1}{2}L_{\max}$ for this example.

3. GROUP ROUND ROBIN ALGORITHM

GRR uses a binary grouping strategy to organize flows into groups of similar weight values which can be more easily scheduled. It then combines two scheduling algorithms: an inter-group scheduling algorithm to select a group from which to select a flow to service, and an intra-group scheduling algorithm to select a flow from within the selected group to service. The two algorithms operate hierarchically and make their decisions independently. Table 2 presents a list of GRR-specific terminology we use. The algorithm can then be briefly described in three parts:

1. **Flow grouping strategy**: Flows are separated into groups of flows with similar weight values. Each group G is assigned flows with weight value between 2^{σ_G} to $2^{\sigma_G+1}-1$, where $\sigma_G \geq 0$ is called the order of group G. Thus, a flow S_j is inserted into group G where $\sigma_G = \lfloor \log_2 \phi_j \rfloor$.

2. **Inter-group scheduling**: An existing scheduling algorithm is used to select a group from which to select a flow to service. A group is selected based on the group weight. Each group acts as a composite flow with respect to the inter-group scheduler.

3. **Intra-group scheduling**: Once a group has been selected, a flow within the group is selected for service using a specialized round-robin algorithm.

Table 2: GRR Terminology

g	The number of groups.
$G(i)$	The group to which S_i belongs.
σ_G	The order of G.
$B_G(t)$	The set of backlogged flows in group G, where $B_G(t)$ stays fixed during the interval (t_1,t_2).
$\phi_G(t)$	The group weight: $\sum_{S_i \in B_G(t)} \phi_i$.
$W_G(t_1,t_2)$	The amount of traffic served for group G: $\sum_{S_i \in G} W_i(t_1,t_2)$.
$w_G(t_1,t_2)$	The normalized traffic for G: $\frac{W_G(t_1,t_2)}{\phi_G}$.
$F_{G,H}(t_1,t_2)$	The relative fairness of groups G and H: $\lvert w_G(t_1,t_2) - w_H(t_1,t_2)\rvert$.
$F_G(t_1,t_2)$	The absolute fairness of group G: $\lvert w_G(t_1,t_2) - w(t_1,t_2)\rvert$.
$E_G(t_1,t_2)$	The absolute service error of group G: $\phi_G F_G(t_1,t_2)$.
$F_{i,G}(t_1,t_2)$	The group-relative fairness of flow S_i: $\lvert w_i(t_1,t_2) - w_G(t_1,t_2)\rvert$.
$E_{i,G}(t_1,t_2)$	The group-relative error of flow S_i: $\phi_i F_{i,G}(t_1,t_2) = \lvert W_i - \frac{\phi_i}{\phi_G} W_G\rvert$.
D_i(t)	The deficit of S_i at time t.

The grouping strategy limits the number of groups that need to be scheduled since the number of groups grows at worst logarithmically with the largest flow weight value. If g is the number of groups, then $g \leq \lfloor \log_2 \phi_{\max} \rfloor + 1$ where $\phi_{\max}$ is the largest possible weight in the system. Even a very large 32-bit flow weight would limit the number of groups to no more than 32. As a result, the inter-group scheduler never needs to schedule a large number of groups which limits the impact of skewed weight distributions on groups. The grouping strategy also limits the weight distributions that the intra-group scheduler needs to consider since the range of weight values within a group is less than a factor of two. As a result, the intra-group scheduler never needs to schedule flows with skewed weight distributions since the flows within a group must have relatively similar weight values.

While GRR is designed to leverage existing scheduling algorithms for inter-group scheduling, GRR takes advantage of its grouping strategy by using a specialized deficit round-robin algorithm for intra-group scheduling to provide good fairness and time complexity bounds. Compared to DRR, the GRR intra-group scheduler has two important differences. First, all flow weights in a group G are normalized with respect to the minimum possible weight, $\phi_{\min} = 2^{\sigma_G}$, for the purpose of service allocation. Second, since normalized flow weights may be non-integers, GRR can provide fractional packet allocations that are accumulated as part of a flow's deficit.

The GRR intra-group algorithm considers the scheduling of flows in rounds. A *round* is one pass through a group's queue of flows from beginning to end. The group queue of flows does not need to be sorted in any manner. During each round, the GRR intra-group algorithm considers the flows in round-robin order. For each backlogged flow S_i, the scheduler serves a maximum of $(\frac{\phi_i}{\phi_{\min}})L_{\max} + D_i(r-1)$ bits. $\phi_{\min}$ is the minimum possible weight in the group. $D_i(r)$, the deficit of flow S_i after round r, is defined recursively as

$$D_i(r) = (\frac{\phi_i}{\phi_{\min}})L_{\max} + D_i(r-1) - \sum_{k=k_i^{r-1}+1}^{k_i^r} L(p_i^k),$$

where k_i^r is the number of S_i packets that left the server up to

and including round r, with $D_i(0) = 0$. Thus, in each round, S_i is allotted $(\frac{\phi_i}{\phi_{\min}})L_{\max}$ bits plus any additional leftover from the previous round, and $D_i(r)$ keeps track of the amount of service that S_i missed because the packet at the head of the flow's queue was too large to fit in the remaining allotted space. We observe that $D_i(r) < L_{i,\max}$ after any round r. It is important to note that the intra-group allocation of service and the deficit of each flow are transparent to the inter-group scheduler, which sees the group as a composite flow. Internally, the intra-group scheduler uses the described mechanism to forward a packet from among its flows whenever the integrity scheduler decides to send a packet from the group. Thus, even when a flow will send more than one packet during a round, it will in general not send them consecutively. Consider the following exampleto demonstrate further how GRR scheduling works. Suppose we use WFQ for the inter-group scheduler in the GRR framework. Consider a set of six flows that need to be scheduled, one flow S_1 with weight 12, two flows S_2 and S_3 each with weight 3, and the other three flows S_4, S_5, and S_6 each with weight 2. Assume that all flows are backlogged and the size of all the packets is $L_{\max} = 1$ for simplicity. The six flows will be put into two groups G_1 and G_2 as follows: $B_{G_1} = \{S_2, S_3, S_4, S_5, S_6\}$ and $B_{G_2} = \{S_1\}$. The weight of the groups are $\phi_{G_1} = 12$ and $\phi_{G_2} = 12$. WFQ will consider the groups in this order: $G_1, G_2, G_1 \ldots G_2$ will schedule flow S_1 every time G_2 is considered for service since it has only one flow. We will show the order of service for the first two rounds of G_1. Rounds 3 and 4 of G_1 have the same order of service as rounds 1 and 2. In beginning of round 1 in G_1, each flow starts with 0 deficit and gains $\frac{\phi_i}{\phi_{\min}(G_1)}$, where $\phi_{\min}(G_1) = 2$. The maximum service that flows S_2, S_3, S_4, S_5, and S_6 can receive in round 1 are 1.5, 1.5, 1, 1, and 1, respectively. Since packets have to be transmitted as a unit, the scheduler will serve 1 packet from each flow in G_1 during round 1. After the first round, the deficit for S_2, S_3, S_4, S_5, and S_6 are 0.5, 0.5, 0, 0, and 0. In the beginning of round 2, each flow gets another $\frac{\phi_i}{\phi_{\min}(G_1)}$ allocation, and the maximum allowed service for S_2, S_3, S_4, S_5, and S_6 becomes 2, 2, 1, 1, and 1. S_2 sends two packets, followed by two packets from S_3 and one packet each from S_4, S_5 and S_6. After round 2, the deficit of all the flows in G_2 becomes 0. The sequence of packets that the scheduler serves is $\{p_2^1, p_1^1, p_3^1, p_1^2, p_4^1, p_1^3, p_5^1, p_1^4, p_6^1, p_1^5, p_2^2, p_1^6, p_2^3, p_1^7, p_3^2, p_1^8, p_3^3, p_1^9, p_4^2, p_1^{10}, p_5^2, p_1^{11}, p_6^2, p_1^{12}\}$, where p_i^k is the k^{th} packet to have arrived for flow S_i.

Dynamic Considerations: Whenever a flow arrives or departs the router, the inter-group scheduler needs to adjust the state of the containing group to account for its new weight. In addition, the inter-group scheduler updates the group's round robin list. A newly arrived flow S_i is added in $G(i)$'s list at the head of the queue, so that the flow will wait until the next round to be serviced. In effect, we assume flows arrive only at round boundaries. We allocate the flow an initial deficit to compensate it for this delay. A departed flow is simply removed from the list. However, we need to distinguish between departed flows and flows that become non-backlogged. The latter should not be placed at the head of the round robin queue if they become backlogged again within the first round, so that they don't receive undue service. Once a flow becomes non backlogged, we remove it from the list, but remember its predecessor. This allows us to reinsert the flow correctly in the list if new packets arrive, so that the inserted flow takes the place of its predecessor. If a full round passes and we reach the predecessor flow, the predecessor links pointing to it are invalidated and the corresponding non-backlogged flows are effectively considered as having departed.

4. GRR FAIRNESS AND TIME COMPLEXITY

We present the worst-case fairness bounds of GRR and then discuss its time complexity. We show that GRR can achieve both good fairness and low service error. We describe GRR fairness in terms of the commonly used relative fairness and absolute service error measures. GRR limits service error by dividing up the scheduling problem into intra-group scheduling and inter-group scheduling. We first analyze the fairness of the intra-group algorithm then discuss this in the context of inter-group scheduling. We first prove that GRR intra-group scheduling provides $O(1)$ relative fairness and absolute service error bounds. We then continue our analysis to prove that GRR scheduling overall provides absolute service error bounded by a constant factor plus the absolute service error of the inter-group scheduling algorithm used.

We start with the relative fairness of the intra-group algorithm and show that the algorithm provides $O(1)$ relative fairness between any two flows. We present the following lemma.

Lemma 1. *The relative fairness $F_{i,j}$ of the GRR intra-group round-robin scheduling algorithm between any two backlogged flows S_i and S_j of the same group is bounded as follows:*

$$F_{i,j} < 2(\frac{L_{\max}}{\phi_i} + \frac{L_{\max}}{\phi_j}).$$

PROOF. For any interval (t_1, t_2) when S_i is continuously backlogged, let r_1^i be the round before the round when S_i is first considered after time t_1, and r_2^i the last round before time t_2 when S_i sends. Then S_i receives during (t_1, t_2) a total service equal to $(r_2^i - r_1^i)(\frac{\phi_i}{\phi_{\min}})L_{\max} + D_i(r_1^i) - D_i(r_2^i)$. Assume without loss of generality that S_j is after S_i in the group. Then $r_1^i - 1 \leq r_1^j \leq r_1^i$ and $r_2^i - 1 \leq r_2^j \leq r_2^i$. Subtracting, we get $r_2^i - r_1^i - 1 \leq r_2^j - r_1^j \leq r_2^i - r_1^i + 1$ or $|(r_2^i - r_1^i) - (r_2^j - r_1^j)| \leq 1$. We also observe that $|D_j(r_1^j) - D_j(r_2^j)| < L_{\max}$ for any j, since $0 \leq D_j(t) < L_{\max}$ at any time t. It then follows that the relative fairness $F_{i,j}$ between flows S_i and S_j is: $F_{i,j}$= $|w_i(t_1, t_2) - w_j(t_1, t_2)|$= $|\{(r_2^i - r_1^i)(\frac{L_{\max}}{\phi_{\min}}) + \frac{(D_i(r_1^i) - D_i(r_2^i))}{\phi_i}\}$ - $\{(r_2^j - r_1^j)(\frac{L_{\max}}{\phi_{\min}}) + \frac{(D_j(r_1^j) - D_j(r_2^j))}{\phi_j}\}| \leq |\{(r_2^i - r_1^i) - (r_2^j - r_1^j)\}(\frac{L_{\max}}{\phi_{\min}})| + |\frac{(D_i(r_1^i) - D_i(r_2^i))}{\phi_i}| + |\frac{(D_j(r_1^j) - D_j(r_2^j))}{\phi_j}| \leq \frac{L_{\max}}{\phi_{\min}} + \frac{L_{\max}}{\phi_i} + \frac{L_{\max}}{\phi_j}$.

Because $\phi_{\min} \leq \phi_i, \phi_j < 2\phi_{\min}$, we can simplify this to either of the following looser bounds: $F_{i,j} \leq \frac{3L_{\max}}{\phi_{\min}}$, or $F_{i,j} < 2(\frac{L_{\max}}{\phi_i} + \frac{L_{\max}}{\phi_j})$. □

Note that if we only consider full rounds, then the bounds on $F_{i,j}$ would be $\frac{L_{\max}}{\phi_i} + \frac{L_{\max}}{\phi_j}$.

We now consider the group-relative service error of the intra-group algorithm and show a stronger result, namely that the algorithm provides $O(1)$ service error. We present the following lemma.

Lemma 2. *The group-relative service error $E_{i,G}$ of the GRR intra-group round-robin scheduling algorithm for flow S_i is bounded as follows:*

$$E_{i,G} < 5L_{\max}.$$

PROOF. Consider the group-relative fairness $F_{i,G}$ of a flow $S_i \in G$: $F_{i,G}(T_0, T) = |w_i(T_0, T) - w_G(T_0, T)|$. To simplify the analysis, we will first consider $F_{i,G}$ in between rounds. Let t_k be the

time round k finishes, and let $t_0 = T_0$ and $t_r = T$. Then during round k, W_j increases by $(\frac{\phi_j}{\phi_{\min}})L_{\max} + D_j(t_{k-1}) - D_j(t_k)$ for any flow S_j in G that is backlogged during that round. We can then write $w_G(t_{k-1}, t_k) = \frac{\sum_{S_j \in B_G(t_{k-1}+0)} W_j(t_{k-1}, t_k)}{\phi_G(t_{k-1}+0)} =$
$\frac{\sum_{S_j \in B_G(t_{k-1}+0)} \{(\frac{\phi_j}{\phi_{\min}})L_{\max} + D_j(t_{k-1}) - D_j(t_k)\}}{\phi_G(t_{k-1}+0)} = \frac{L_{\max}}{\phi_{\min}} +$
$\frac{D_G(t_{k-1}+0) - D_G(t_k - 0)}{\phi_G(t_{k-1}+0)}$,
where we denote by $D_G(t)$ the sum of the deficits of the backlogged flows in G at time t. We make the distinction between $D_G(t_k - 0)$ and $D_G(t_k + 0)$ since after a round, we may adjust $D_G(t)$ by assigning some initial deficit to a new flow that arrived during that round. We can have three situations after each round:

1. The backlog set B_G remains unchanged (no session arrived or became idle). Then $D_G(t_k - 0) = D_G(t_k + 0)$ and $\phi_G(t_{k-1} + 0) = \phi_G(t_k + 0)$.

2. A new session S_{new} arrived: $\phi_G(t_k + 0) = \phi_G(t_{k-1} + 0) + \phi_{new}$. We place the new session at the beginning of the queue (so that it does not run during the current round) and after the round, we assign it a deficit of $\frac{\phi_{new} D_G(t_k - 0)}{\phi_G(t_{k-1}+0)}$. Then, $D_G(t_k + 0) = D_G(t_k - 0) + \frac{\phi_{new} D_G(t_k - 0)}{\phi_G(t_{k-1}+0)} = \frac{D_G(t_k - 0)\phi_G(t_k+0)}{\phi_G(t_{k-1}+0)}$. Thus, $\frac{D_G(t_k-0)}{\phi_G(t_{k-1}+0)} = \frac{D_G(t_k+0)}{\phi_G(t_k+0)}$.

3. A session S_{dep} departed (became idle): $\phi_G(t_k+0) = \phi_G(t_{k-1}+0) - \phi_{dep}$. In this case, the work that the departing flow executes during the round is less than its quota. To keep the analysis simple, we can assume for the purpose of keeping track of fairness that the departing flow has received $D_{dep}(t_{k-1}) + \frac{\phi_{dep}}{\phi_{\min}} L_{\max} - \phi_{dep} \frac{D_G(t_k+0)}{\phi_G(t_k+0)})$ which is less than $D_{dep}(t_{k-1}) + \frac{\phi_{dep}}{\phi_{\min}} L_{\max}$ and positive (since $\frac{D_G(t_k+0)}{\phi_G(t_k+0)} \leq \frac{|B_G(t_k+0)|L_{\max}}{|B_G(t_k+0)|\phi_{\min}} = \frac{L_{\max}}{\phi_{\min}}$). Then the normalized work done for round k is $\frac{L_{\max}}{\phi_{\min}} +$
$\frac{D_G(t_{k-1}+0) - \left(D_G(t_k-0) + \frac{\phi_{dep} D_G(t_k+0)}{\phi_G(t_k+0)}\right)}{\phi_G(t_{k-1}+0)}$. Since we make no changes to D_G after round k, $D_G(t_k+0) = D_G(t_k-0)$ and we have a normalized service of
$\frac{L_{\max}}{\phi_{\min}} + \frac{D_G(t_{k-1}+0) - \frac{(\phi_G(t_k+0)+\phi_{dep}) D_G(t_k+0)}{\phi_G(t_k+0)}}{\phi_G(t_{k-1}+0)} =$
$\frac{L_{\max}}{\phi_{\min}} + \frac{D_G(t_{k-1}+0)}{\phi_G(t_{k-1}+0)} - \frac{D_G(t_k+0)}{\phi_G(t_k+0)}$.

Of course, we can have any number of sessions arrive and depart during the same round, in which case the results in situations 2 and/or 3 still apply by superposition. Thus, in all three cases, the normalized work during a round can be written as $\frac{L_{\max}}{\phi_{\min}} + \frac{D_G(t_{k-1}+0)}{\phi_G(t_{k-1}+0)} - \frac{D_G(t_k+0)}{\phi_G(t_k+0)}$, so $F_{i,G}(T_0, T) = |\sum_{k=1}^{r} (\frac{1}{\phi_i}[\frac{L_{\max}\phi_i}{\phi_{\min}} + D_i(t_{k-1}) - D_i(t_k)]) - \sum_{k=1}^{r} [\frac{L_{\max}}{\phi_{\min}} + \frac{D_G(t_{k-1}+0)}{\phi_G(t_{k-1}+0)} - \frac{D_G(t_k+0)}{\phi_G(t_k+0)}]| =$
$|\frac{D_i(t_0) - D_i(t_r)}{\phi_i} - \frac{D_G(t_0+0)}{\phi_G(t_0+0)} + \frac{D_G(t_r+0)}{\phi_G(t_r+0)}|$. We now make use of the fact that $0 \leq D_G(t) \leq |B_G(t)|L_{\max} \leq (\frac{\phi_G(t)}{\phi_{\min}})L_{\max}$ and $0 \leq D_i \leq L_{\max}$ to get $F_{i,G}(T_0, T) \leq |\frac{D_i(T_0) - D_i(T)}{\phi_i}| + |\frac{D_G(T_0)}{\phi_G(T_0)} - \frac{D_G(T)}{\phi_G(T)}| \leq \frac{L_{\max}}{\phi_i} + \frac{L_{\max}}{\phi_{\min}} < \frac{L_{\max}}{\phi_i} + \frac{2L_{\max}}{\phi_i} = \frac{3L_{\max}}{\phi_i}$ since $\phi_{\min} > \frac{\phi_i}{2}$. The group-relative error $E_{i,G}$ in between rounds is then bounded by $E_{i,G}(T_0, T) \leq 3L_{\max}$. Since during a round, a flow's normalized work w_i can get ahead or behind by at most an additional $\frac{L_{\max}}{\phi_{\min}}$, in general $F_{i,G}(T_0, T) < \frac{5L_{\max}}{\phi_i}$ and $E_{i,G}(T_0, T) < 5L_{\max}$ for any time interval (T_0, T) when S_i is continuously backlogged. □

Given the $O(1)$ relative fairness and absolute service error bounds for the GRR intra-group scheduling algorithm, we can now analyze the overall GRR algorithm. We first show the following theorem that states the GRR algorithm relative fairness is bounded by the relative fairness of the inter-group scheduler plus a constant factor.

Theorem 1. *The relative fairness $F_{i,j}$ of the GRR scheduling algorithm between any two flows S_i and S_j is bounded by a constant plus the relative fairness between the respective groups containing the flows as follows:*

$$F_{i,j} < \frac{5L_{\max}}{\phi_i} + \frac{5L_{\max}}{\phi_j} + F_{G(i),G(j)}.$$

PROOF. For any two flows S_i and S_j in groups $G(i)$ and $G(j)$ and continuously backlogged in the interval (t_1, t_2), we have:
$F_{i,j}(t_1, t_2) = |w_i(t_1, t_2) - w_j(t_1, t_2)| = |w_i(t_1, t_2) - w_{G(i)}(t_1, t_2) + w_{G(i)}(t_1, t_2) - w_{G(j)}(t_1, t_2) + w_{G(j)}(t_1, t_2) - w_j(t_1, t_2)| \leq$
$|w_i(t_1, t_2) - w_{G(i)}(t_1, t_2)| + |w_{G(j)}(t_1, t_2) - w_j(t_1, t_2)| +$
$|w_{G(i)}(t_1, t_2) - w_{G(j)}(t_1, t_2)| \leq F_{i,G(i)} + F_{j,G(j)} + F_{G(i),G(j)}$.
Since the group-relative fairness of flows is bounded by the results of Lemma 2, it follows that the relative fairness of the overall GRR algorithm between any two flows is bounded by $F_{i,j}(t_1, t_2) < \frac{5L_{\max}}{\phi_i} + \frac{5L_{\max}}{\phi_j} + F_{G(i),G(j)}$. Of course, if both flows are part of the same group, then $F_{i,j} < \frac{2L_{\max}}{\phi_i} + \frac{2L_{\max}}{\phi_j}$. This is a special case, but one that would occur frequently in practice, where many flows tend to have the same or similar weights. □

Combining intra-group and inter-group error bounds in a similar manner, we can show the following theorem which states the stronger result that the overall GRR algorithm service error is bounded by the service error of the inter-group scheduler plus a constant factor.

Theorem 2. *The absolute service error E_i of the GRR scheduling algorithm of flow S_i is bounded by a constant plus the absolute service error of the inter-group scheduling algorithm as follows:*

$$E_i < 5L_{\max} + E_G.$$

PROOF. Let flow S_i be backlogged in group G during the interval (T_0, T). Then $F_i(T_0, T) = |w_i(T_0, T) - w(T_0, T)| = |w_i(T_0, T) - w_G(T_0, T) + w_G(T_0, T) - w(T_0, T)| \leq |w_i(T_0, T) - w_G(T_0, T)| + |w_G(T_0, T) - w(T_0, T)| = F_{i,G}(T_0, T) + F_G(T_0, T)$. The overall error will be at most $E_i(T_0, T) = \phi_i F_i(T_0, T) \leq E_{i,G}(T_0, T) + \phi_i F_G(T_0, T) = E_{i,G}(T_0, T) + (\frac{\phi_i}{\phi_G})E_G(T_0, T) \leq E_{i,G}(T_0, T) + E_G(T_0, T)$. We have shown that $E_{i,G} < 5L_{\max}$ for the intra-group round robin scheduler. Thus, the overall error will be on the order of the overall error of the inter-group scheduler used: $E_i(T_0, T) < 5L_{\max} + E_G(T_0, T)$. □

Having analyzed the fairness of GRR, we now analyze GRR **time complexity**. Using GRR, scheduling the next packet to transmit entails choosing the group and picking the appropriate flow from within the group that gets to send a packet. The time for the GRR intra-group scheduler to select a flow for service from a group is $O(1)$. This follows from the fact that the round robin always considers backlogged flows in the same order, and serves each with at least one packet since $L(p_i^k) \leq L_{\max}$ and $\frac{\phi_i}{\phi_{\min}} \geq 1$. Non-backlogged flows are removed from the queues so that the scheduler does not waste any time looping through flows with no packets to send. All the list removal and insertion steps involved in the dynamic operation are $O(1)$. Since selecting a flow is $O(1)$ for the intra-group round-robin algorithm, the overall complexity will be the same as that of the inter-group scheduler.

5. USING GRR WITH OTHER SCHEDULING ALGORITHMS

We describe how GRR can be used with a number of existing scheduling algorithms by incorporating those algorithms as the GRR inter-group scheduling algorithm. We show how using GRR in conjunction with these algorithms can improve their fairness and service error bounds and reduce their time complexity.

Applying GRR is simple, provided that the algorithm used for inter-group scheduling can handle dynamically changing weights properly. To understand this requirement, we need to consider the dynamic behavior of GRR-based schedulers. Flow arrivals and departures can be handled easily, by simply adding and removing them from the corresponding group round-robin lists. However, this operation also involves adjusting the group's weight accordingly, by adding in or subtracting out the flow's weight. Dynamic weight adjustment in the scheduled entities should therefore not affect the proper operation of the inter-group scheduler. This is in fact true for all the six popular schedulers we consider in this context: WFQ, SCFQ, SFQ, hierarchical stride (HS), WF^{2}Q, and SRR. Virtual time algorithms such as WFQ, WF^{2}Q, SCFQ, and SFQ use weights only when computing packet timestamps, and thus naturally absorb any change in weights. SRR schedules based on its Weight Matrix. As long as the entry in this matrix is updated, weight changes properly result in proportionally adjusted service allocations. for HS, whenever a leaf node (a scheduled entity) changes its weight, the normalized work counters for all the ancestor nodes in the HS-tree need to be updated. Thus, flow arrivals and departures are handled by $O(1)$ operations, except for HS, which requires $O(\log g)$ time.

In the subsections that follow, we discuss the relative fairness and absolute service error bounds for each of the above algorithms. We combine these algorithms with GRR to construct six new scheduling algorithms: GWFQ, GSCFQ, GSFQ, GHS, GWF^{2}Q, and GSRR.

5.1 GWFQ

WFQ [9, 14] is a virtual time fair queuing algorithm. WFQ emulates GPS by serving packets in the order in which they would finish service under GPS. WFQ introduced virtual finishing times (VFT) for this purpose and services the flow with the packet with the earliest VFT.

For WFQ, the relative fairness between any two flows S_i and S_j is bounded by $\frac{L_{\max}}{\phi_i} + \frac{L_{\max}}{\phi_j}$ as shown by Lemma 3 in the Appendix. (2) then implies that the absolute service error of WFQ is bounded by $NL_{\max}$, where N is the number of flows being scheduled. We can see that the service error bound is not less than $O(N)$ by considering the example of $N+1$ flows, the first with weight N and the rest weight 1. If all packets are of size $L_{\max}$, WFQ will service N packets from the first flow before servicing any other flows, resulting in service error of $\frac{NL_{\max}}{2}$. Because WFQ must order the flows based on their VFTs, the time complexity of scheduling is $O(\log N)$. We can combine WFQ with GRR to derive a new scheduling algorithm GWFQ that has both a lower service error bound and lower time complexity. GWFQ is simply the GRR scheduler using WFQ as the inter-group scheduling algorithm. GWFQ only uses WFQ for scheduling among groups, reducing the number of entities that are considered by the WFQ algorithm. This reduces service error and time complexity since the service error bound and time complexity of WFQ grow with the number of entities being scheduled.

For GWFQ, the relative fairness between any two flows S_i and S_j is bounded by $6(\frac{L_{\max}}{\phi_i} + \frac{L_{\max}}{\phi_j})$, as computed based on Theorem 1. This is an upper bound but may not be a tight bound. Theorem 2 shows that the absolute service error is a constant factor plus the absolute service error of the inter-group scheduler. Since the WFQ inter-group scheduler has absolute service error $gL_{\max}$, where g is the number of groups, the absolute service error of GWFQ is $gL_{\max} + 5L_{\max}$, which is $O(g)$. The time complexity of GWFQ is $O(\log g)$. If we assume that the number of groups is bounded by a constant, this reduces to a $O(1)$ bound on absolute service error and time complexity.

5.2 GSCFQ

SCFQ is a virtual time scheduling algorithm that avoids the overhead of keeping track of an ideal GPS server by setting the system virtual time to be the virtual finishing time of the packet being serviced. For SCFQ, the relative fairness between any two flows S_i and S_j has been shown to be bounded by $\frac{L_{\max}}{\phi_i} + \frac{L_{\max}}{\phi_j}$ [10]. [10] did not show a service error bound, but as in the case of WFQ, (2) implies the absolute service error bound of SCFQ is $NL_{\max}$, where N is the number of flows being scheduled. The time complexity of SCFQ scheduling is $O(\log N)$. The basic fairness and time complexity properties of SCFQ are similar to WFQ.

Like GWFQ, we can use SCFQ as the inter-group scheduling algorithm with GRR to derive a new scheduling algorithm GSCFQ that has both a lower service error bound and lower time complexity. Since SCFQ and WFQ have the same relative fairness and service error bounds, GSCFQ has the same relative fairness and service error bounds as GWFQ, namely relative fairness bounded by $6(\frac{L_{\max}}{\phi_i} + \frac{L_{\max}}{\phi_j})$ and absolute service error of $gL_{\max} + 5L_{\max}$. Similarly, the time complexity of GSCFQ is $O(\log g)$.

5.3 GSFQ

SFQ is similar to SCFQ, except that the system virtual time is the start time of the packet in service, and packets are selected based on their virtual start times. For SFQ, the relative fairness between any two flows S_i and S_j has been shown to be bounded by $\frac{L_{\max}}{\phi_i} + \frac{L_{\max}}{\phi_j}$ [12]. [12] did not show a service error bound, but as in the case of WFQ and SCFQ, (2) implies the absolute service error bound of SFQ is $NL_{\max}$, where N is the number of flows being scheduled. The time complexity of SFQ scheduling is $O(\log N)$.

Like GWFQ and GSCFQ, we can use SFQ as the inter-group scheduling algorithm with GRR to derive a new scheduler GSFQ that has both a lower service error bound and lower time complexity. Since SFQ and WFQ have the same relative fairness and service error bounds, GSFQ has the same relative fairness and service error bounds as GWFQ, namely relative fairness bounded by $6(\frac{L_{\max}}{\phi_i} + \frac{L_{\max}}{\phi_j})$ and absolute service error of $gL_{\max} + 5L_{\max}$. Similarly, the time complexity of GSFQ is $O(\log g)$.

5.4 GHS

A different group approach than that of GRR was previously proposed in the context of stride CPU scheduling [19]. We discuss an adaptation of that algorithm for packet scheduling that we refer to as HS. HS arranges flows in a balanced binary tree, where the flows are the leaves, and any internal node behaves with respect to its parent like a flow with a weight equal to the sum of the weights of its children. When an internal node is selected, then it in turn selects one of its children, until a leaf node is reached and serviced. Normalized work counters for the flow and all of its ancestors are incremented with the amount of bits transmitted by the flow. A parent node selects from its children p and q the node whose normalized work is less than the normalized work of the parent. If $w_p = w_q = w$, then some tie-breaking policy can be used, for example, select the node with the higher weight, or select

the left node, etc. The benefit of this hierarchical approach can be most easily illustrated by the case of $N+1$ flows, the first with weight N and the rest weight 1. If all packets are of size $L_{\max}$, a non-hierarchical stride scheduler would service N packets from the first flow before servicing any other flows, resulting in $O(N)$ service error. However, HS aggregates multiple flows so that it ends up interleaving the execution of the weight N flow with the other flows of weight 1, resulting in a smaller $O(\log N)$ service error.

For HS, we prove that the relative fairness between any two flows S_i and S_j is bounded by $\frac{\lceil \log(N) \rceil L_{\max}}{\min(\phi_i, \phi_j)}$ where N is the number of flows being scheduled (Appendix Lemma 5). We also show that the absolute service error of HS is bounded by $\lceil \log(N) \rceil L_{\max}$ (Appendix Lemma 4) Since HS must do a traversal of the balanced binary tree from the root to a leaf, the time complexity of HS scheduling is $O(\log N)$.

The hierarchical grouping strategy of HS provides some of the same benefits of the GRR framework. However, we can combine HS with GRR to derive a new scheduling algorithm GHS that provides even lower service error and time complexity. GHS is simply the GRR scheduler using HS as the inter-group scheduling algorithm. GHS only uses HS for scheduling among groups, reducing the number of entities that are considered by the HS algorithm. This reduces service error and time complexity since the service error bound and time complexity of HS grow with the number of entities being scheduled.

For GHS, the relative fairness between any two flows S_i and S_j is
$F_{i,j}$ ¡$F_{G(i),G(j)}$ + $\frac{5L_{\max}}{\phi_i} + \frac{5L_{\max}}{\phi_j} \leq \frac{(\lceil \log g \rceil) L_{\max}}{\min(\phi_{G(i)}, \phi_{G(j)})} + \frac{10 L_{\max}}{\min(\phi_i, \phi_j)} \leq \frac{L_{\max}(\lceil \log g \rceil + 10)}{\min(\phi_i, \phi_j)}$,
where g is the number of groups of flows being scheduled. The absolute service error of GHS is $E_i \leq E_{i,G(i)} + E_{G(i)} < 5L_{\max} + (\lceil \log g \rceil) L_{\max} = (\lceil \log g \rceil + 5) L_{\max}$. The time complexity of GHS is $O(1) + O(\log g) = O(\log g)$.

Note that the motivation of hierarchy in HS and GRR is to reduce GPS service error bounds and is fundamentally different from the conventional notion of hierarchical packet scheduling algorithms which seek to provide H-GPS fairness [2]. The idea of grouping flows together is also at the heart of these algorithms, but their goal is to emulate H-GPS instead of GPS. The aim is to provide isolation in link sharing and to implement different policy-based service classes. Thus, fairness is provided among the children groups of a node, but not across the entire system, and unused service from a group is distributed solely inside the parent group. However, the idea of using an instance of the same type of virtual time server at each node can also be adapted to simulate GPS, by letting the weights of the group nodes vary when flows enter or leave the groups. Still, such an approach gives error and computational complexity bounds that are a factor of k larger than those for the individual server, where k is the height of the tree, and are not well-suited for our purpose.

5.5 GWF^{2}Q

WF^{2}Q is a virtual time algorithm that is identical to WFQ, except it considers for service only the packets that would have already started service under the equivalent GPS. This difference enables WF^{2}Q to provide the lowest service error of all fair queuing algorithms. For WF^{2}Q, the absolute service error has been shown to be bounded by $L_{\max}$ [1], which is $O(1)$. As a result, (1) implies for WF^{2}Q that its relative fairness between any two flows S_i and S_j is bounded by $(\frac{L_{\max}}{\phi_i} + \frac{L_{\max}}{\phi_j})$. The time complexity of WF^{2}Q scheduling is $O(\log N)$.

We can combine WF^{2}Q with GRR to derive a new scheduling algorithm GWF^{2}Q that preserves the $O(1)$ service error bound while providing lower time complexity. GWF^{2}Q is the GRR scheduler using WF^{2}Q as the inter-group scheduler. GWF^{2}Q only uses WF^{2}Q for scheduling among groups, reducing the number of entities that are considered by the WF^{2}Q algorithm. This reduces time complexity since the time complexity of WF^{2}Q grows with the number of entities being scheduled.

For GWF^{2}Q, the relative fairness between any two flows S_i and S_j is bounded by $6(\frac{L_{\max}}{\phi_i} + \frac{L_{\max}}{\phi_j})$, as computed based on Theorem 1. Theorem 2 shows that the absolute service error is a constant factor plus the absolute service error of the inter-group scheduler. Since the WF^{2}Q inter-group scheduler has absolute service error $L_{\max}$, the absolute service error of GWF^{2}Q is less than $6L_{\max}$, which is $O(1)$. Similarly, the time complexity of GWF^{2}Q is $O(\log g)$. If we assume that the number of groups is constant, this reduces to a $O(1)$ bound on absolute service error and time complexity.

5.6 GSRR

SRR is an improvement on deficit round-robin scheduling that still services flows in a fixed order, but interleaves them to address the burstiness and poor delay properties of round-robin. SRR introduces a Weight Matrix and uses the concept of a k-order Weight Spread Sequence (WSS), where k is the number of bits needed to store the weight of the flows. The Weight Matrix consists of binary vectors coded from the weights of the flows. SRR then scans the elements of the Weight Matrix in a fixed order specified by WSS and selects the flow to execute whose weight corresponds to the matrix element selected.

For SRR, the relative fairness between any two flows S_i and S_j has been shown to be bounded by $\frac{(k+2)L_{\max}}{2\min(\phi_i, \phi_j)}$ [7]. The absolute service error of SRR is $E_i \leq \frac{k(N+1)}{2} L_{\max} = O(Nk)$, as stated in Lemma 6, which we prove in the Appendix 9. The time complexity of SRR scheduling is $O(1)$.

We can combine SRR with GRR to derive a new scheduling algorithm GSRR that can preserve the $O(1)$ time complexity of SRR while providing a lower service error bound. GSRR is simply the GRR scheduler using SRR as the inter-group scheduling algorithm. However, since the weight of a group increases with the number of flows, we cannot have a bound for k as assumed in SRR, so we cannot use a pre-computed WSS. We can have SRR simulate in real time the WSS, allowing theoretically for any size weights, at the expense of computational complexity.

For GSRR, the relative fairness between any two flows S_i and S_j is bounded by $\frac{(k+22)L_{\max}}{2\min_{S_p \in (G(i) \cup G(j))}(\phi_p)}$, where $k = \log_2(\phi_{G,\max})$ and $\phi_{G,\max}$ is the largest group weight. In worst case, k is $O(\log(N\phi_{\max})) = O(\log N + \log \phi_{\max}) = O(\log N + g)$. The absolute service error of SRR is $E_i < 5L_{\max} + \frac{(g+1)k}{2} L_{\max} = O(g(\log N + g))$. Applying the grouping strategy to SRR thus reduces the error from $O(kN)$ to $O(g(\log N + g))$ where $g \leq k = \log \phi_{\max}$, while maintaining the strict $O(1)$ time complexity.

6. DELAY BOUNDS FOR GRR

We show in the following that the good fairness worst-case bounds on GRR translate into good bounds on the delay experienced by packets, which make the very efficient GRR algorithm well-suited for guaranteeing QoS in high-speed networks. We start by demonstrating the worst-case delay bounds for GRR, and show how they compare against the metrics of other algorithms. Since we cannot predict network usage, we give our delay bounds without making any assumption about the incoming traffic envelope, or other flow control characteristics.

We define the delay of a packet p_i^k as $d(p_i^k) - a(p_i^k)$, which is the time spent by the packet in the scheduler queue. For any scheduler, we can express the delay as $\frac{W(d(p_i^k),a(p_i^k))}{R}$, since during the interval $(d(p_i^k), a(p_i^k))$ the scheduler is busy and is serving at rate R. Since the delay naturally depends on the queue size that the packet sees upon arrival, it equals

$$\begin{aligned} & \frac{\phi}{R}\left(\frac{W_i(d(p_i^k),a(p_i^k))}{\phi_i} + \frac{W(d(p_i^k),a(p_i^k))}{\phi} - \frac{W_i(d(p_i^k),a(p_i^k))}{\phi_i}\right) \\ = \quad & \frac{\phi}{R}\left(\frac{Q_i(a(p_i^k))}{\phi_i} + w(d(p_i^k), a(p_i^k)) - w_i(d(p_i^k), a(p_i^k))\right) \\ \leq \quad & \frac{\phi}{R}\left(\frac{Q_i(a(p_i^k))}{\phi_i} + F_i(d(p_i^k), a(p_i^k))\right). \end{aligned}$$

If the server guarantees rate R_i for flow S_i, then, since $R_i \leq \frac{\phi_i}{\phi} R$, we can bound $d(p_i^k) - a(p_i^k)$ by $\frac{Q_i(a(p_i^k))}{R_i} + \frac{\phi_i}{R_i} F_i(d(p_i^k), a(p_i^k))$. The second term shows the excess delay incurred by the packet beyond the time taken to empty the queue seen by the incoming packet under the worst-case guaranteed rate and represented by the first term.

In the case of GRR, we can write, based on Theorem 2,

$$\begin{aligned} d(p_i^k) - a(p_i^k) \quad & \leq \\ \frac{Q_i(a(p_i^k))}{R_i} + \frac{\phi_i}{R_i}\left(\frac{5L_{\max}}{\phi_i} + F_{G(i)}(d(p_i^k), a(p_i^k))\right) \quad & \leq \\ \frac{Q_i(a(p_i^k))}{R_i} + \frac{5L_{\max}}{R_i} + \frac{\phi_{G(i)}}{R_i} F_{G(i)}(d(p_i^k), a(p_i^k)). & \end{aligned}$$

We can then consider the resulting delay bounds when GRR is used with other scheduling algorithms discussed in Section 5.

For WF^{2}Q, $F_i \leq \frac{L_{i,\max}}{\phi_i} + \frac{L_{\max} - L_{i,\max}}{\phi}$, and thus the excess delay is bounded by $\frac{L_{i,\max}}{R_i} + \frac{L_{\max} - L_{i,\max}}{R}$. Using GRR with WF^{2}Q, the extra delay is therefore bounded by $\frac{5L_{\max}}{R_i} + \frac{L_{G(i),\max}}{R_{G(i)}} + \frac{L_{\max} - L_{G(i),\max}}{R}$. If $L_{G(i),\max} = L_{i,\max}$, and since $\phi_{G(i)} \geq \phi_i$, the bound becomes

$$\frac{6L_{\max}}{R_i} + \frac{L_{\max} - L_{i,\max}}{R}.$$

Thus, GRR only introduced a constant factor of 6 in the $O(1)$ delay of WF^{2}Q.

For WFQ, SCFQ, and SFQ, from (2) we can obtain $F_i \leq \frac{L_{i,\max}}{\phi_i} + \frac{(N-1)L_{\max}}{\phi}$, and thus the excess delay is bounded by $\frac{L_{i,\max}}{R_i} + \frac{(N-1)L_{\max}}{R}$. Using GRR with WF^{2}Q, the extra delay is instead bounded by

$$\frac{5L_{\max}}{R_i} + \frac{\phi_i}{R_i}\frac{L_{G(i),\max}}{\phi_{G(i)}} + \frac{(g-1)L_{\max}}{R}.$$

If $L_{G(i),\max} = L_{i,\max}$, and since $\phi_{G(i)} \geq \phi_i$, the bound becomes $\frac{6L_{\max}}{R_i} + \frac{(g-1)L_{\max}}{R}$. In these cases, GRR improves the $O(N)$ delay to only $O(g)$ delay. It is instructive to consider the two components of this extra delay. The term containing $\frac{L_{\max}}{R_i}$ is inversely proportional to the flow guaranteed rate, so that flows with large weights are not very affected by it. On the other hand, the term containing $\frac{L_{\max}}{R}$ is independent of the flow under consideration, invariably affecting any flows, including flows with large weights. It is therefore highly desirable to limit the multiplicative factor for this term. GRR achieves this goal by preserving the factor of 1 unchanged in the case of WF^{2}Q, and improving it from $N-1$ to $g-1$ for the other algorithms.

SRR, like most other round-robin schedulers, has poor delay bounds. GSRR manages to decrease the delay in this case as well, from $\frac{(k+2)}{R_i} + \frac{(k+2)(N-1)}{R}$ to

$$\frac{(g + \log N)}{R_i} + \frac{(g + \log N)(g-1)}{R},$$

where $k \leq g$ is defined in the discussion on GSRR. While the first term increases to a $\log N$ factor, the second, weight independent term, gets decreased significantly.

It is shown in [20] that $O(N)$ delay bounds are unavoidable for scheduling algorithms of less than $O(\log N)$ time complexity. This result has been misused to claim optimal delay properties on some low complexity schedulers that incur $O(N)$ extra delay, where delay is defined as above. We point out that [20] refers to a different measure of delay, known as GPS-relative delay, defined for packet p_i^k as $d(p_i^k) - d^{GPS}(p_i^k)$. This measure considers the time a packet leaves the server compared to a reference GPS server, and is of very little practical relevance. Instead, for the delay measure we used, it is in fact possible to have $O(1)$ delay and $O(1)$ time complexity, as we have shown for GRR combined with WF^{2}Q.

7. EXPERIMENTAL RESULTS

To show the performance of GRR in practice, we present some results that quantify the resulting service error for various combinations of weights and flows for some of the algorithms presented in Section 5. We compare the service errors of WFQ, WF^{2}Q, and SRR against their respective GRR counterparts, GWFQ, GWF^{2}Q, and GSRR. For this purpose, we developed a scheduling simulator to examine the scheduling behavior of these different algorithms across hundreds of thousands of different combinations of flows with different weight values.

The simulator takes four inputs, the scheduling algorithm, the number of flows N, the total number of weights ϕ, and the number of flow-weight combinations. The simulator randomly assigns weights to flows and scales the weight values to ensure that they add up to ϕ. It then schedules the flows using the specified algorithm and tracks the resulting service error. The simulator runs the scheduler until the resulting schedule repeats, then computes the maximum (most positive) and minimum (most negative) service error across the non-repeating portion of the schedule for the given set of flows and weight assignments. This process of random weight allocation and scheduler simulation is repeated for the specified number of flow-weight combinations. We then compute an average maximum service error and average minimum service error for the specified number of flow-weight combinations to obtain an "average-case" error range, as well as the overall maximum and minimum error for a given (N, ϕ) pair.

Since the absolute service error of a scheduler is often most clearly illustrated with skewed weight distributions, we ran simulations for each scheduling algorithm considered on 35 different combinations of N and ϕ, with one of the flows given a weight equal to 10 percent of ϕ. All of the other flows were then randomly assigned weights to sum to the remaining 90 percent of ϕ. For each set of (N, ϕ), we ran 2500 flow-weight combinations and determined the resulting average and extreme values of the minimum and maximum error ranges. For simplicity, all simulations were run with all flows backlogged at all times and all packets of equal size. The average service error ranges normalized by packet size for WFQ, WF^{2}Q, SRR, GWFQ, GWF^{2}Q, and GSRR with these skewed weight distributions are shown in Figures 1 to 6. Each figure shows four surfaces, two representing the average maximum and minimum service error and two representing the overall extreme values for all client share combinations as a function of N and ϕ for the respective scheduling algorithm. Note that the standard deviation of the errors measured across the 2500 iterations is small, and the average and extreme error surfaces overlap for most (N, ϕ) pairs.

Figure 1 shows the service error range for WFQ to be large, ranging between -1 to 819. WFQ has a lower bound of -1 but no

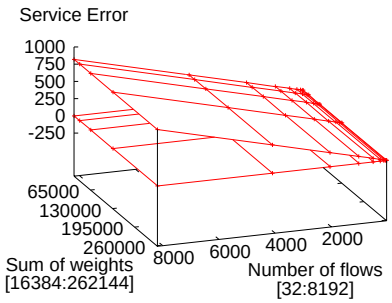

Figure 1: WFQ service error

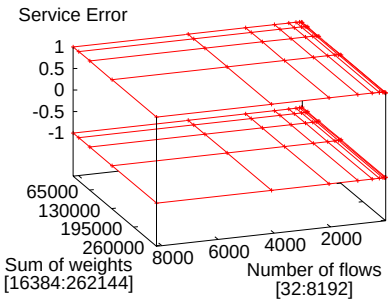

Figure 2: WF^{2}Q service error

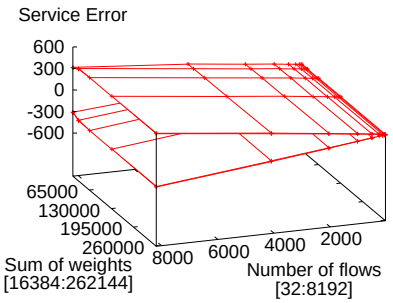

Figure 3: SRR service error

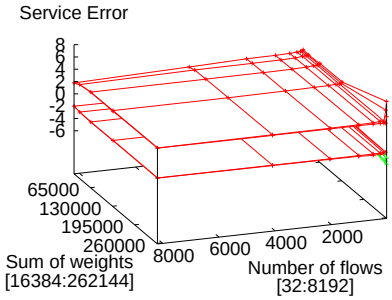

Figure 4: GWFQ service error

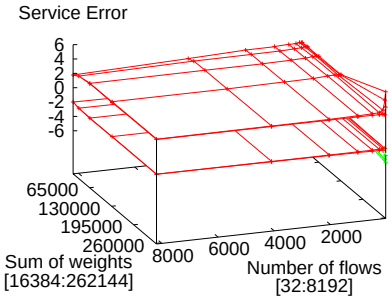

Figure 5: GWF^{2}Q service error

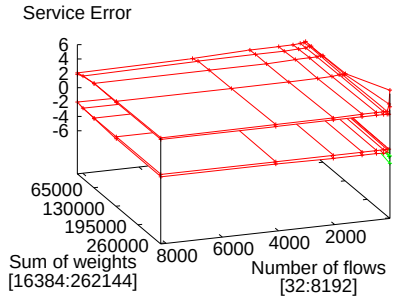

Figure 6: GSRR service error

constant upper bound on the error. In comparison, Figure 4 shows that GWFQ has significantly less service error than WFQ, ranging only from -4.21 to 6.01 while also preserving a constant lower bound. Figure 2 shows the service error range for WF^{2}Q, which is provably bounded between -1 to 1. In comparison, Figure 5 shows that GWF^{2}Q has only slightly larger service error ranging between -4.24 to 5.64, which is within the derived GWF^{2}Q constant service error bounds and is achieved with lower time complexity. Figure 3 shows the service error for SRR to be quite large, ranging between -374 to 374. In comparison, Figure 6 shows that GSRR has much less service error than SRR, ranging only between -4.33 to 5.78. These results quantitatively demonstrate the benefits GRR can provide in improving service error with often lower time complexity.

8. RELATED WORK

Many fair packet scheduling algorithms have been developed [9, 22, 12, 10, 17, 8, 16, 7]. These algorithms can be loosely classified as time-stamp-based algorithms and round-robin algorithms. We have shown that GRR complements this work and can be applied to these classes of algorithms to improve their fairness and time complexity. As shown in the Appendix, we also prove several new results regarding the relative fairness and absolute service error bounds for previously proposed scheduling algorithms, such as WFQ [9], Hierarchical Stride [19], and SRR [7].

GRR's grouping strategy was also used in Stratified Round Robin (StRR) [15], but in a less general way. StRR is based on distributing flows into classes, where all flows with weights between 2^{-k} and $2^{-(k-1)}$ belong to class F_k. StRR splits time into scheduling slots and then makes sure to assign all the flows in class F_k one slot every scheduling interval, which is defined to consist of 2^k consecutive slots for class F_k. To account for the variation of up to a factor of two of the weights of the flows within a class, the algorithm has an intra-class aspect to it, allotting a credit of $2^k \phi_i L_{\max}$ to flow $S_i \in F_k$ for each slot it is assigned, with a deficit keeping track of unused service. This is also similar to GRR's intra-group solution, with the key difference that a flow can send all of its packets during a round consecutively, while in GRR, a flow is allowed to send only one packet every time its group is selected, regardless of its deficit or round allocation. In spite of its higher $O(g)$ scheduling complexity, StRR has weaker fairness guarantees than GRR-based schedulers: its service error may grow as $O(N)$, a consequence of its deficit accounting scheme ([4]). Note also that StRR bounds the extra delay of a packet p_i^k, $d(p_i^k) - a(p_i^k) - \frac{Q_i(a(p_i^k))}{R_i}$, by $\frac{5L_{\max}}{R_i} + 5(N-1)\frac{L_{\max}}{R}$. Comparing the $\frac{6L_{\max}}{R_i} + 5(g-1)\frac{L_{\max}}{R}$ delay bounds of GWFQ, GSCFQ, and GSFQ with the delay of StRR, we note the significant decrease from $O(N)$ to only $O(g)$. GWF^{2}Q achieves a strict $O(1)$ bound.

Finally, unlike StRR, GRR provides a scheduling framework that can combined with existing scheduling algorithms to improve their properties. This provides a much more generalized grouping strategy that can offer a broad range of GRR-based algorithm choices with different fairness and time complexity.

In this context, [21] illustrates an application of GRR to the WF^{2}Q algorithm. The properties found by the authors of [21] are consistent with the GRR framework and its analysis.

9. CONCLUSIONS

We have presented the design and analysis of Group Round-Robin, a packet scheduler that combines a binary grouping strategy with a specialized deficit round-robin algorithm to improve fairness and reduce time complexity. GRR is designed to utilize existing algorithms as the inter-group scheduler for scheduling among its groups. We proved that GRR only adds a constant factor to the relative fairness and absolute service error of any inter-group scheduler and also has low time complexity. As a result, we showed that GRR can be used to reduce the service error and time complexity of virtual time algorithms such as WFQ, SCFQ, SFQ, maintain the constant service error bound of WF^{2}Q while reducing its time complexity, and reduce the service error of SRR while maintaining its constant time complexity. Furthermore, we prove for the first time the fairness and service error bounds for several of these algo-

rithms, enabling us to compare these approaches using consistent fairness measures in a manner that has not been previously done. We implemented GRR and several GRR augmented algorithms and showed for various flows and weight distributions that GRR can reduce the service error of existing algorithms such as WFQ and SRR by well more than an order of magnitude. GRR's ability to narrow the trade-off between fairness and computational complexity provides an effective packet scheduling mechanism for data networks.

Acknowledgments

Chris Vaill developed the scheduling simulator used for our experiments. This work was supported in part by NSF grants CNS-0426623 and ANI-0240525.

10. REFERENCES

[1] J. Bennett and H. Zhang, "WF^2Q: Worst-case Fair Weighted Fair Queuing," in *Proceedings of INFOCOM '96*, San Francisco, CA, Mar. 1996.

[2] J. Bennett and H. Zhang, "Hierarchical Packet Fair Queuing Algorithms," in *Proceedings of ACM SIGCOMM '96*, Aug. 1996.

[3] B. Caprita, W. Chan and J. Nieh, "Group Round Robin: Improving the Fairness and Complexity of Packet Scheduling." Tech. Rep. CUCS-018-03, Department of Computer Science, Columbia University, June 2003.

[4] B. Caprita, W. C. Chan, J. Nieh, C. Stein, and H. Zheng, "Group-Ratio Round Robin: O(1) Proportional Share Scheduling for Uniprocessor and Multiprocessor Systems," in *Proceedings of the 2005 USENIX Annual Technical Conference*, USENIX, Anaheim, CA, April 2005.

[5] W. Chan and J. Nieh, "Group ratio round-robin: An O(1) proportional share scheduler." Tech. Rep. CUCS-012-03, Department of Computer Science, Columbia University, April 2003.

[6] S. Cheung and C. Pencea, "BSFQ: Bin Sort Fair Queuing," in *Proceedings of INFOCOM ' 02*, New York, NY, June 2002.

[7] G. Chuanxiong, "SRR: An 0(1) Time Complexity Packet Scheduler for Flows in Multi-Service Packet Networks," in *Proceedings of ACM SIGCOMM '01*, Aug. 2001.

[8] J. Cobb, M. Gouda, and A. El-Nahas, "Time-Shift Scheduling - Fair Scheduling of Flows in High-Speed Networks," in *IEEE/ACM Transactions on Networking*, June 1998.

[9] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queuing Algorithm," in *Proceedings of ACM SIGCOMM '89*, Austin, TX, Sept. 1989.

[10] S. J. Golestani, " A Self-Clocked Fair Queuing Scheme for Broadband applications," in *Proceedings of IEEE INFOCOM '94*, Apr. 1994.

[11] P. Goyal, S. Lam, and H. Vin, "Determining End-to-End Delay Bounds in Heterogeneous Networks," in *Proceedings NOSSDAV*, Apr. 1995.

[12] P. Goyal, H. Vin, and H. Cheng, "Start-Time Fair Queuing: A Scheduling Algorithm for Integrated Services Packet Switching Networks," in *IEEE/ACM Transactions on Networking*, Oct. 1997.

[13] L. Kleinrock, *Queuing Systems, Volume II: Computer Applications*. New York: John Wiley & Sons, 1976.

[14] A. Parekh and R. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case," in *IEEE/ACM Transactions on Networking*, 1(3), June 1993.

[15] S. Ramabhadran and J. Pasquale, "Stratified Round Robin: A Low Complexity Packet Scheduler with Bandwidth Fairness and Bounded Delay," in *Proceedings of ACM SIGCOMM '03*, Karlsruhe, Germany, August 2003.

[16] M. Shreedhar and G. Varghese, "Efficient Fair Queuing Using Deficit Round-Robin," in *Proceedings of ACM SIGCOMM '95*, 4(3), Sept. 1995.

[17] D. Stiliadis, and A. Varma, "Efficient Fair Queuing Algorithms for Packet-Switched Networks," in *IEEE/ACM Transactions on Networking*, Apr. 1998.

[18] I. Stoica, H. Abdel-Wahab, and K. Jeffay, "On the Duality between Resource Reservation and Proportional Share Resource Allocation," in *Multimedia Computing and Networking Proceedings, SPIE Proceedings Series*, 3020, Feb. 1997.

[19] C. Waldspurger, "Lottery and Stride Scheduling: Flexible Proportional-Share Resource Management". PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Sept. 1995.

[20] J. Xu and R. Lipton, "On Fundamental Tradeoffs between Delay Bounds and Computational Complexity in Packet Scheduling Algorithms," in *Proceedings of ACM SIGCOMM '02*, Pittsburgh, PA, August 2002.

[21] X. Yuan and Z. Duan, "FRR: a Proportional and Worst-Case Fair Round Robin Scheduler," in *Proceedings of INFOCOM '05*, Miami, FL , Mar. 2005.

[22] L. Zhang, "Virtual Clock: A New Traffic Control Algorithm for Packet Switched Networks," in *ACM Transactions on Computer Systems*, 9(2), May 1991.

APPENDIX

Many scheduling algorithms have been developed and analyzed, but often times these results have been incomplete in considering both relative fairness and absolute service error bounds. Some algorithms have only been analyzed in terms of relative fairness while others have only been analyzed in terms of absolute service error. The use of different fairness measures makes it difficult to objectively compare different scheduling algorithms. The following theorems and lemmas provide new rigorous analysis supported by detailed proofs that complement previously known results and offer for the first time a consistent basis for comparing different scheduling algorithms.

As discussed in Section 2, relative fairness is a widely used measure. In that context, a theorem in [10] states that if a packet-based algorithm guarantees $F_{i,j}(t_1, t_2) \leq M_{i,j}$ for any interval (t_1, t_2) when S_i and S_j are backlogged, then $M_{i,j}$ must be at least $\frac{1}{2}(\frac{L_{\max}}{\phi_i} + \frac{L_{\max}}{\phi_j})$. $M_{i,j}$ is a theoretical bound on the relative fairness which usually depends on the weights of the flows and the packet size. While this result is often cited, no proof has been shown. We state and prove a tighter result for relative fairness given by the following theorem.

Theorem 3. *For some scheduler, let* $M_{i,j} = \sup\{F_{i,j}(t_1, t_2)\}$ *where the supremum is taken over any packet arrival pattern for the flows, and over any interval* (t_1, t_2) *such that* S_i, $S_j \in B(t)$, $\forall t \in (t_1, t_2)$. *Then*

$$M_{i,j} \geq \frac{L_{i,\max}}{\phi_i} + \frac{L_{j,\max}}{\phi_j} - \epsilon,$$

where $\epsilon = \frac{1}{f} \min\{\frac{L_{i,\max}}{\phi_i}, \frac{L_{j,\max}}{\phi_j}\}$ *if* $\exists e, f \in \mathbf{Z}_+$ *such that*

$\gcd(e,f)=1$ *such that* $\frac{L_{i,\max}\phi_j}{L_{j,\max}\phi_i}=\frac{e}{f}$ *(i.e.,*$\frac{L_{i,\max}\phi_j}{L_{j,\max}\phi_i}\in\mathbf{Q}$*), and* $\epsilon=0$ *otherwise.*

PROOF. To simplify the notation, let us define $\Delta_{i,j}(t_1,t_2)=\frac{W_i(t_1,t_2)}{\phi_i}-\frac{W_j(t_1,t_2)}{\phi_j}$. Then, $F_{i,j}=|\Delta_{i,j}|$.

We will consider the particular arrival pattern when both S_i and S_j are backlogged at any time $t\geq 0$, and all packets in S_k are of size $L_{k,\max}$, $k\in\{i,j\}$.

Let $x_k=\frac{L_{k,\max}}{\phi_k}$, $k\in\{i,j\}$. Without loss of generality, assume $x_i\geq x_j$.

Let $\delta>0$ be an arbitrarily small real. Also, denote by A the set $\{(cx_j-dx_i|c,d\in\mathbf{Z}_+,cx_j-dx_i>0\}$.

Then $\exists m\in A$, $m=c_mx_j-d_mx_i$ such that

$$0<m<\inf A+\delta. \tag{3}$$

Now let t_0 be such that $\Delta_{i,j}(0,t_0)<\inf_{t\geq 0}\{\Delta_{i,j}(0,t)\}+m$. Then we have $\Delta_{i,j}(t_0,t)=\Delta_{i,j}(0,t)-\Delta_{i,j}(0,t_0)>-m$, $\forall t\geq t_0$. We can write $\Delta_{i,j}(t_0,t)$ in the form $n_i(t)x_i-n_j(t)x_j$, with $n_k(t)\in\mathbf{Z}_+$ denoting the number of packets of flow S_k ($k\in\{i,j\}$) sent in the interval (t_0,t). Then since $\Delta_{i,j}(t_0,t)>-m$,

$$n_i(t)x_i-n_j(t)x_j>-m. \tag{4}$$

Let t be such that at time t, packet d_m+1 of S_i is scheduled for sending. Then after the packet is sent, $\Delta_{i,j}(t_0,t+\frac{L_{i,\max}}{R})$ will be $x_i+n_i(t)x_i-n_j(t)x_j$, where $n_i(t)=d_m$. If $n_i(t)x_i-n_j(t)x_j\geq x_j$, then $\Delta_{i,j}\geq x_i+x_j$, and our proof is done. Otherwise, $\Delta_{i,j}=x_i+x_j-((n_j(t)+1)x_j-d_m(t)x_i)$, and $0<(n_j(t)+1)x_j-d_m(t)x_i<x_j+m$. The second inequality follows from (4). We will show it is necessary that $n_j(t)+1=c_m$. We know $m=c_mx_j-d_mx_i$ satisfies (3). If $c_m<n_j(t)+1$, then $c_m+1\leq n_j(t)+1$, and so $(n_j(t)+1)x_j-d_m(t)x_i\geq m+x_j$, contradiction. If $c_m>n_j(t)+1$, then $c_m\geq n_j(t)+2$ and so $m\geq(n_j(t)+1)x_j-d_m(t)x_i+x_2>x_2$. Since $\inf A<x_2$ and δ can be set arbitrarily small, this contradicts (3).

Therefore, $\Delta_{i,j}=x_i+x_j-m$. By finding an upper bound ϵ on $\inf A$, we can show $\Delta_{i,j}\geq x_i+x_j-(\inf A+\delta)\geq x_i+x_j-\epsilon-\delta$. Since δ can be made arbitrarily small, we can show a $x_i+x_j-\epsilon$ lower bound on $\Delta_{i,j}$. This will imply a lower bound on $F_{i,j}$.

We distinguish two cases:

1. a $\frac{x_i}{x_j}\in\mathbf{Q}$ i.e., $\exists e,f\in\mathbf{Z}_+$, $\gcd(e,f)=1$ such that $\frac{x_i}{x_j}=\frac{e}{f}$. Then $\epsilon=\inf\{x_j(c-d\frac{x_i}{x_j})|c,d\in\mathbf{Z}_+,x_j(cx_j-dx_i)>0\}=\frac{x_j}{f}\inf\{(cf-de|c,d\in\mathbf{Z}_+,cf-de>0\}$. We observe that $I=\inf\{cf-de|c,d\in\mathbf{Z}_+,cf-de>0\}=\min\{cf-de|c,d\in\mathbf{Z}_+,cf-de>0\}=c_If-d_Ie$ is just $\gcd(f,e)$. Indeed, since $\gcd(f,e)|e$ and $\gcd(f,e)|f$, then $\gcd(f,e)|(c_Ie-d_If)=I$, and so $\gcd(f,e)\leq I$. Also, we know that $\forall e,f\in\mathbf{Z}_+,\exists\alpha,\beta\in\mathbf{Z}$ such that $e\alpha+f\beta=\gcd(f,e)$. If $\alpha<0$ and $\beta>0$, consider $I-\gcd(e,f)=(c_I-\alpha)e-(d_I+\beta)f$. Since $\alpha<0$ and $\beta>0$, $c_I-\alpha>0$ and $d_I+\beta>0$, and so $I-\gcd(e,f)\in\{(cf-de|c,d\in\mathbf{Z}_+,cf-de>0\}$. But then we would have $I\leq I-\gcd(e,f)$, impossible. Therefore, $\alpha>0$ and $\beta<0$ and thus $\gcd(e,f)=e\alpha-f(-\beta)\in\{(cf-de|c,d\in\mathbf{Z}_+,cf-de>0\}$, and so $I\leq\gcd(e,f)$. Thus, since $\gcd(f,e)\leq I$ and $I\leq\gcd(e,f)$, $\gcd(e,f)=I$. But $\gcd(e,f)=1$, and so $I=1$. Thus, $\epsilon=\frac{x_j}{f}I=\frac{x_j}{f}$.
2. b $\frac{x_i}{x_j}\in\mathbf{R}-\mathbf{Q}$. Then we show $\epsilon=0$. Otherwise, let $k=\min\{k\in\mathbf{Z}|k\epsilon\geq x_j\}$. By the choice of k, we have $(k-1)\epsilon<x_j$ so
$$\exists\tau>0\ \textit{such that}\ (k-1)\epsilon=x_j-\tau. \tag{5}$$
Now let $c,d\in\mathbf{Z}_+$ be such that $\epsilon+\frac{\tau}{k}>cx_j-dx_i\geq\epsilon$. Multiplying the double inequality by k, and subtracting x_j, we obtain
$$\epsilon k+\tau-x_j>kcx_j-kdx_i-x_j\geq\epsilon k-x_j. \tag{6}$$
Since $\epsilon k-x_j\geq 0$, we have $kcx_j-kdx_i-x_j=(kc-1)x_j-kdx_i\geq 0$. In fact, the inequality is strict. Otherwise, if $(kc-1)x_j-kdx_i=0$, we would have $\frac{x_i}{x_j}=\frac{kc-1}{kd}\in\mathbf{Q}$, contradiction. Therefore, $(kc-1)x_j-kdx_i>0$, so that $(kc-1)x_j-kdx_i\in A$. By the definition of ϵ, $(kc-1)x_j-kdx_i\geq\epsilon$.
We use this in (6) to get $\epsilon k+\tau-x_j>\epsilon$ or $(k-1)\epsilon>x_j-\tau$, contradicting (5).

□

WFQ has been analyzed in terms of absolute service error, but no proofs have been given showing its relative fairness bounds. We state and prove the following result for WFQ.

Lemma 3. *For WFQ, the relative fairness* $F_{i,j}$ *between any two flows* S_i *and* S_j *is bounded by*

$$\frac{L_{max}}{\phi_i}+\frac{L_{max}}{\phi_j}.$$

PROOF. We first make the important observation that for servers that reference GPS, such as WFQ, a flow is assumed backlogged for the purpose of fairness if it is backlogged under GPS. The distinction is important, since otherwise, we can easily construct an example where a flow backlogged only under WFQ receives half the service as another flow with the same weight. Consider any two flows S_i and S_j GPS-backlogged during the interval (t_1,t_2). If we denote by $f_n(t)$ the virtual finishing time of the packet at the head of $Q_{n,n\in\{i,j\}}$, then

$$f_i(t)-f_j(t)\leq\frac{L_{i,max}}{\phi_i} \tag{7}$$

Clearly, this difference is maximal just after a packet p_i^k from S_i is served, and stays fixed until another packet from S_i or S_j is served. Before p_i^k departed, $f_i(d(p_i^k)-0)<f_j(d(p_i^k)-0)$. Also, $f_i(d(p_i^k)+0)=f_i(d(p_i^k)-0)+\frac{L(p_i^{k+1})}{\phi_i}$ and $f_j(d(p_i^k)-0)=f_j(d(p_i^k)+0)$. It then follows that $f_i(d(p_i^k)+0)<f_j(d(p_i^k)+0)+\frac{L(p_i^{k+1})}{\phi_i}$, which proves (7). We observe that $f_n(t_1)=f_v(p_n^{k_1^n})$ and $f_n(t_2)=f_v(p_n^{k_2^n})$ where by $p_n^{k_m^n}$ we denote the last packet to have departed from flow S_n before time t_m.

Because for any packet arrival in the backlog interval, the virtual finishing time is $f_v(p_n^k)=f_v(p_n^{k-1})+\frac{L(p_n^k)}{\phi_n}$, $n\in\{i,j\}$, we have $f_v(p_i^{k_2^i})=f_v(p_i^{k_1^i})+w_i(t_1,t_2)$ and $f_v(p_i^{k_2^j})=f_v(p_i^{k_1^j})+w_j(t_1,t_2)$. Subtracting, we get $w_i(t_1,t_2)-w_j(t_1,t_2)=f_v(p_i^{k_2^i})-f_v(p_i^{k_1^i})-(f_v(p_j^{k_2^j})-f_v(p_j^{k_1^j}))=f_i(t_2)-f_j(t_2)+f_j(t_1)-f_i(t_1)\leq\frac{L_{i,max}}{\phi_i}+\frac{L_{i,max}}{\phi_j}$. Similarly, $w_j(t_1,t_2)-w_i(t_1,t_2)\leq\frac{L_{i,max}}{\phi_i}+\frac{L_{i,max}}{\phi_j}$. It follows that $F_{i,j}(t_1,t_2)=|w_i(t_1,t_2)-w_j(t_1,t_2)|\leq\frac{L_{i,max}}{\phi_i}+\frac{L_{i,max}}{\phi_j}$. □

HS has been proposed as an approach to improve scheduling fairness, but no formal analysis has been previously done showing its relative fairness and absolute service error bounds.

To facilitate comparison with other scheduling approaches, we state and prove the following lemmas that bound the relative fairness and absolute service error of HS.

Lemma 4. *For HS, the absolute service error E_i of flow S_i is bounded as follows:*

$$E_i \leq (k-1)L_{\max},$$

where $k = \lceil \log(N) \rceil + 1$.

PROOF. Let k be the height of the HS tree. Define $p^0 = S_i$, which is a leaf node, and p^j = parent(p^{j-1}) for $j = 1, \ldots, k-1$. For some $j \leq k-2$, $w_{p^j} - w_{p^{j+1}} = \frac{\phi_q(w_{p^j} - w_q)}{(\phi_{p^j} + \phi_q)}$ where q is the sibling of p^j. Since p^{j+1} always selects the child with the smaller normalized work, at any time, $w_{p^j} - w_q \leq \frac{L_{\max}}{\phi_{p^j}}$ and $w_q - w_{p^j} \leq \frac{L_{\max}}{\phi_q}$. Using $\phi_q = \phi_{p^{j+1}} - \phi_{p^j}$, we get $\frac{-L_{\max}}{\phi_{p^j+1}} \leq w_{p^j} - w_{p^{j+1}} \leq \frac{L_{\max}}{\phi_{p^j}} - \frac{L_{\max}}{\phi_{p^j+1}}$. Since $w_i = w_{p^1}$ and $w = w_{p^{k-1}}$, we have $w_i - w = \sum_{j=0}^{k-2}(w_{p^j} - w_{p^{j+1}})$ which is then bounded above by $\sum_{j=0}^{k-2}(\frac{L_{\max}}{\phi_{p^j}} - \frac{L_{\max}}{\phi_{p^j+1}}) = L_{\max}(\frac{1}{\phi_i} - \frac{1}{\phi})$. Also, $w_i = w_{p^1} \geq \sum_{j=0}^{k-2}(-\frac{L_{\max}}{\phi_{p^j+1}}) \geq -L_{\max}\sum_{j=0}^{k-3}\frac{1}{\phi_{p^1}} + \frac{L_{\max}}{\phi} \geq -\frac{(k-1)L_{\max}}{\phi_i}$, since $\phi_{p^i} \geq \phi_{p^1}$. Thus, the error $\phi_i(w_i - w)$ can get only as large as $L_{\max}$, and cannot be less than $-(k-1)L_{\max}$. Thus, $E_i = O(k) = O(\log N)$.

We will now show that this bound is tight. Consider the case with a flow S_1 of weight N and N flows of weight 1, and assume fixed packet sizes L for simplicity. The first time, S_1 is selected, and the work of the leaf p^0 corresponding to S_1 as well as the work of p^j, $j = 1, \ldots, k-1$ will become L. The next $k-1$ packets will be served from the siblings of p^j, $j = 0, \ldots, k-2$ since the $w_{p^j} > 0 = w_q$ where q is the sibling of p^j. Thus, after the first k packets have been served, the error of S_1 will be $|1 - \frac{k\phi_1}{\phi}| = |1 - \frac{kN}{2N}| = |1 - \frac{k}{2}|$. Thus, the error is worst case $\Omega(k) = \Omega(\log N)$. □

Lemma 5. *For HS, the relative fairness $F_{i,j}$ is bounded by*

$$\frac{(k-1)L_{\max}}{\min(\phi_i, \phi_j)} \leq \frac{\lceil \log(N) \rceil L_{\max}}{\min(\phi_i, \phi_j)},$$

where N is the number of flows being scheduled and $k = \lceil \log(N) \rceil + 1$.

PROOF. Assuming without loss of generality that $w_i \geq w_j$, using the results in Lemma 4, we have
$F_{i,j} = w_i - w_j = w_i - w + w - w_j \leq L_{\max}(\frac{1}{\phi_i} - \frac{1}{\phi}) + \frac{(k-2)L_{\max}}{\phi_j} + \frac{L_{\max}}{\phi} \leq \frac{(k-1)L_{\max}}{\min(\phi_i, \phi_j)}$. □

SRR has been previously analyzed in terms of its relative fairness bounds, but no formal analysis has been previously done showing its absolute service error bounds. To facilitate comparison with other scheduling approaches, we state and prove the following lemma that bounds the absolute service error of SRR.

Lemma 6. *For SRR, the absolute service error E_i of flow S_i is bounded as follows:*

$$E_i \leq \frac{(N+1)k}{2} = O(Nk),$$

where N is the number of flows being scheduled and k is the number of bits needed to store the maximum weight of any flow.

PROOF. We will first show a $O(Nk)$ bounds for E_i of SRR. For this, we introduce the equivalent complete binary tree associated with the WSS. This tree has k levels, and each node on level j corresponds to the WSS number j (where 'k' is the root and '1' are the leaves). We notice that the inorder traversal of this tree corresponds to the k^{th} order WSS (this observation allows us to devise a $O(1)$ algorithm to dynamically generate the WSS). In the following discussion, we assume $L = L_{\max} = 1$. The error bounds can be scaled after-wards by $L_{\max}$.

Consider flow S_i with weight ϕ_i. Since the leaves in the WSS-tree are visited every other time, we can assume that the node currently visited is a leaf node (otherwise, the error can increase by at most 1). Let β be the number of leaves considered thus far. Then the number of level j nodes visited is $\lfloor \frac{\beta}{2^{j-1}} + \frac{1}{2} \rfloor$, as can be deduced from the WSS-tree. Then the work done for flow S_i is $W_i = \sum_{j=1}^{k} N_j^i \lfloor \frac{\beta}{2^{j-1}} + \frac{1}{2} \rfloor$ while the total work done is $W = \sum_{j=1}^{k} N_j \lfloor \frac{\beta}{2^{j-1}} + \frac{1}{2} \rfloor$ where N_j is the number of flows that have a non-zero bit at position j, and N_j^i is the j^{th} bit of ϕ_i. Thus, $W_i - (\frac{\phi_i}{\phi})W = \sum_{j=1}^{k} N_j^i \lfloor \frac{\beta}{2^{j-1}} + \frac{1}{2} \rfloor - \frac{\sum_j N_j^i 2^{k-j}}{\sum_j N_j 2^{k-j}} \sum_j N_j \lfloor \frac{\beta}{2^{j-1}} + \frac{1}{2} \rfloor$. Then $W_i - \frac{\phi_i}{\phi}W \leq \sum_j N_j^i(\frac{\beta}{2^{j-1}} - \frac{1}{2}) - \frac{\sum_j N_j^i 2^{-j}}{\sum_j N_j 2^{-j}} \sum_j N_j(\frac{\beta}{2^{j-1}} + \frac{1}{2}) = \frac{\sum_j N_j^i}{2} + \frac{\sum_j N_j^i 2^{-j}}{\sum_j N_j 2^{-j}} \frac{\sum_j N_j}{2} \leq \frac{k}{2} + \frac{Nk}{2}$, since $\sum_j N_j^i \leq k$ and $\sum_j N_j \leq Nk$ (each of ϕ_i can have at most k bits on). Also, $W_i - (\frac{\phi_i}{\phi})W \geq \sum_j N_j^i(\frac{\beta}{2^{j-1}} + \frac{1}{2}) - \frac{\sum_j N_j^i 2^{-j}}{\sum_j N_j 2^{-j}} \sum_j N_j(\frac{\beta}{2^{j-1}} - \frac{1}{2}) = -\frac{\sum_j N_j^i}{2} - \frac{\sum_j N_j^i 2^{-j}}{\sum_j N_j 2^{-j}} \frac{\sum_j N_j}{2} \geq \frac{-k}{2} - \frac{Nk}{2}$. Therefore, $E_i = |W_i - (\frac{\phi_i}{\phi})W| \leq \frac{k(N+1)}{2} = O(Nk)$.

We will now show that in fact E_i can be $\Omega(kN)$. For this, consider the example with $N+1$ flows, S_1 through S_N, where ϕ_1 = 11...1 101...010$_2$ = $2^k - \frac{2^{\frac{k}{2}}+2}{3}$ and $\phi_{i,i>1}$ = 00...0 010...101$_2$ = $\frac{2^{\frac{k}{2}}-1}{3}$ (we assume k is a multiple of 4). The bit representations of ϕ_i satisfy the following: the first $\frac{k}{2}$ bits are 1 for ϕ_1 and 0 for $\phi_{i,i>1}$, while bit $j > \frac{k}{2}$ is $j \bmod 2$ for ϕ_1 and $(j+1) \bmod 2$ for $\phi_{i,i>1}$, where bit 1 is the most significant bit. Let p be the level $\frac{k}{2}$ node in the WSS-tree such that in the path from the root to p, odd level nodes are right children, and even level nodes are left children. For each node, because of the inorder traversal of the WSS-tree, the left subtree will have been visited before the node, and the right subtree is visited after the node. Thus, at the moment we are considering, only the even level nodes in the path and their left subtrees will have been visited. The following relation holds for the work done during a completely visited subtree rooted at a node on level $\frac{k}{2} + j$, j odd: $W_1\phi - W\phi_1 < 2^k\frac{N}{3} - 2^{\frac{k}{2}+j}\frac{N}{3}$. ($W_1\phi - W\phi_1 = W_1(\phi - \phi_1) - (W - W_1)\phi_1 = ((2^{\frac{k}{2}+j}) - \frac{(2^j+1)}{3})((2^{\frac{k}{2}-1})\frac{N}{3}) - ((2^j - 1)\frac{N}{3})(2^k - \frac{(2^{\frac{k}{2}}+2)}{3}) = 2^{k+j}\frac{N}{3} - 2^{\frac{k}{2}+j}\frac{N}{3} - 2^{\frac{k}{2}+j}\frac{N}{9} - 2^j\frac{N}{9} - 2^{\frac{k}{2}}\frac{N}{9} + \frac{N}{9} - 2^{k+j}\frac{N}{3} + 2^k\frac{N}{3} + 2^{\frac{k}{2}+j}\frac{N}{9} - 2^{\frac{k}{2}}\frac{N}{9} - \frac{2N}{9} = 2^k\frac{N}{3} - 2^{\frac{k}{2}+j}\frac{N}{3} + 2^j\frac{N}{3} - 2^{\frac{k}{2}+1}\frac{N}{9} - \frac{N}{9} \leq 2^k\frac{N}{3} - 2^{\frac{k}{2}+j}\frac{N}{3}$).

Then for all the completely visited subtrees up to the time node p is considered, the difference $W_1\phi - W\phi_1$ is less than $\sum_{j=1}^{\frac{k}{2}-1}(2^k\frac{N}{3}) - \sum_{j=1}^{\frac{k}{2}-1}(2^{\frac{k}{2}+j}\frac{N}{3}) = k2^{k-1}\frac{N}{3} - (2^{k+1} - 2^{\frac{k}{2}+1})\frac{N}{3}$. The S_1 error at node p is then $E_1 = W_1 - (\frac{\phi_1}{\phi})(W + N\frac{k}{2}) = \frac{W_1\phi - W\phi_1 - \phi_1 N\frac{k}{2}}{\phi} \leq (\frac{1}{\phi})(k2^{k-1}\frac{N}{3} - (2^{k+1} - 2^{\frac{k}{2}+1})\frac{N}{3} - (2^k - \frac{2^{\frac{k}{2}}+2}{3})k\frac{N}{2}) \leq (\frac{1}{\phi})(N2^{k-1})(\frac{k}{3} - k) \leq -(\frac{1}{\phi})(\frac{k2^kN}{3}) \leq -\frac{(k2^k\frac{N}{3})}{2^k - \frac{(2^{\frac{k}{2}}+2)}{3} + \frac{N(2^{\frac{k}{2}}-1)}{3}} \leq -\frac{Nk}{6}$ provided that $N \leq 2^{k/2}$. Clearly, if $N > 2^{k/2}$, then we can use the $N, 1, 1, \ldots, 1$ (N times) example to show that the error is $\Omega(N)$, which is unacceptably large by itself. □