A Proportionate Fair Scheduling Rule with Good Worst-case Performance*

Micah Adler
Dep't of Computer Science
University of Massachusetts,
Amherst, USA
micah@cs.umass.edu

Petra Berenbrink
Tom Friedetzky
School of Computing Science
Simon Fraser University,
Burnaby, Canada
(petra,tkf)@cs.sfu.ca

Leslie Ann Goldberg
Paul Goldberg
Mike Paterson
Dep't of Computer Science
University of Warwick,
Coventry, UK
(leslie,pwg,msp)@dcs.warwick.ac.uk

ABSTRACT

In this paper we consider the following scenario. A set of n jobs with different threads is being run concurrently. Each job has an associated weight, which gives the proportion of processor time that it should be allocated. In a single time quantum, p threads of (not necessarily distinct) jobs receive one unit of service, and we require a rule that selects those p threads, at each quantum. Proportionate fairness means that over time, each job will have received an amount of service that is proportional to its weight. That aim cannot be achieved exactly due to the discretisation of service provision, but we can still hope to bound the extent to which service allocation deviates from its target. It is important that any scheduling rule be simple since the rule will be used frequently.

We consider a variant of the Surplus Fair Scheduling (SFS) algorithm of Chandra, Adler, Goyal, and Shenoy. Our variant, which is appropriate for scenarios where jobs consist of multiple threads, retains the properties that make SFS empirically attractive but allows the first proof of proportionate fairness in a multiprocessor context. We show that when the variant is run, no job lags more than $pH(n) - p + 1$ steps below its target number of services, where $H(n)$ is the Harmonic function. Also, no job is over-supplied by more than $O(1)$ extra services. This analysis is tight and it also extends to an adversarial setting, which models some situations in which the relative weights of jobs change over time.

*Supported by NSF Research Infrastructure Award EIA-0080119, NSF Faculty Early Career Development Award CCR-0133664, and the Future and Emerging Technologies Programme of the EU under contract number IST-1999-14186 (ALCOM-FT).

SPAA 2003 San Diego, California USA

1. INTRODUCTION

In this paper, we consider the problem of scheduling a set of n jobs on p processors, where the objective is to schedule the jobs so that, at every time step in the schedule, each job has received as close to a proportionate share of scheduling slots as possible. Assuming that time is discretised into time steps (or quanta), a scheduler may, in each step, allocate processors to p out of the n jobs. If job i has a weight w_i associated with it, *proportionate fairness* requires that after t steps, the number of times that job i has been assigned a processor should be close to $t \cdot p \cdot w_i / \sum_j w_j$.

A very strict sense of this kind of fairness is called *P-Fairness* [3], which requires that after t steps, job i has been assigned to a processor for either $\lfloor (t \cdot p \cdot w_i / \sum_j w_j) \rfloor$ or $\lceil (t \cdot p \cdot w_i / \sum_j w_j) \rceil$ time steps. In other words, every job receives as close to its proportion of service as is possible given integral service constraints. P-Fairness was first introduced in [3], and a number of papers have addressed the problem of designing P-Fair scheduling algorithms that are efficient and practical [4, 1, 2, 10, 6].

An important application of proportionate scheduling algorithms is an operating system assigning jobs to quanta on a multi-processor system. There are a number of practical reasons why existing P-Fair algorithms are not ideally suited for this task. For example, these algorithms assume fixed-length quanta (i.e., no job ever blocks in the middle of a quantum), and they assume that there are no arrivals or departures of jobs. Furthermore, in [6] it is shown that when one of the existing P-Fair algorithms is applied to scenarios with both variable-length quanta and arrivals and departures, the schedule becomes non-work-conserving: at some time steps, a processor is left idle even when there are more jobs in the system than processors. Also, existing P-Fair algorithms are still somewhat complicated and, in fact, [1] provides evidence that designing simpler P-Fair algorithms may be quite difficult. However, for a task such as the assignment of jobs to quanta on a multi-processor system, it is crucial that the scheduling algorithm be extremely fast, since the scheduler will be called by the operating system on the expiration of every quantum.

A simple proportionate fair scheduling algorithm is introduced in [5]. This algorithm is called SFS (Surplus Fair

Scheduling), and can be viewed as a generalisation to multiprocessors of scheduling techniques based on generalised processor sharing, which have been well studied for use in uniprocessor systems [9, 7, 11]. In SFS, each job i maintains a quantity S_i, the "start time". For every round where job i is run, S_i is incremented by $\frac{1}{w_i}$. At the start of each round, the p jobs are run that have the minimum values of $\alpha_i = w_i(S_i - v)$, where $v = \min_j S_j$. [5] provides empirical evidence that SFS has good fairness properties. Furthermore, the algorithm is simple, and generalises quite easily to scenarios with both variable-length quanta, as well as arrivals and departures of jobs. Also, the algorithm is always work-conserving. However, despite the experimental evidence that SFS performs well, the task of proving that SFS does in fact always schedule jobs so that each receives close to a proportionate share of the available processing power has remained an open problem. We note that a number of simple schedulers have been introduced for the (easier) uniprocessor case that have provable guarantees on fairness, including [9, 12, 8, 13].

Note that SFS is not P-fair. Initially, all values of S_i are zero. Hence, any job i has $\alpha_i = 0$ before it is run for the first time by a processor. Hence, any job that has not yet been run will have a higher priority than any job that has been run, so in the first approximately n/p steps, all jobs will be run for the first time. However, P-fairness could require high-weight jobs to be run more than once before the low-weight jobs are run. For a high-weight job i at time $t = \lfloor n/p \rfloor$, its target service allocation $t \cdot p \cdot w_i / \sum_j w_j$ could be more than 2 (if $p|n$ this just requires $w_i > \frac{2}{n}\sum_j w_j$).

A version of the SFS algorithm has an intuitively appealing interpretation, which we call the *leaky-bucket* problem; the focus of this paper will be on studying this problem. We think of each job as a bucket of water that leaks. During each time step, k_i units leak from bucket i, and the total volume of leaked water is p. This is replaced by adding p refills of unit size into each of the p emptiest buckets. Initially, all the buckets have A units of water, for some value of A.

For the correspondence between the leaky-bucket problem and SFS, we first redefine the value v of SFS as the weighted average of the start times ($v = \frac{\sum_j w_j S_j}{\sum_j w_j}$) instead of the minimum start time. Note that this new value of v grows at the same rate as with the original definition from [5], and the algorithm maintains all of the properties that make it attractive from a practical point of view. With the new version of v, the value α_i represents the water level in bucket i. During a step of SFS, the value of v increases by $\frac{p}{\sum_j w_j}$, which corresponds to a decrease in each α_i of $k_i = \frac{pw_i}{\sum_j w_j}$ and a total decrease of $\sum_i k_i = p$. On the other hand, each job i that gets run has α_i increased by 1, giving a total increase of p.

The key observation that relates the leaky-bucket problem to fairness guarantees is the following. After t steps, the number of times that job i is assigned a processor is $(tpw_i/\sum_j w_j)$ $+ X_i(t+1) - A$, where $X_i(t+1)$ denotes the current load of bucket i and A denotes the average load. (This is formalized as Observation 1.)

It turns out that the variant of the scheduling protocol described above seems hard to analyze. Hence, we here study a slight modification of the emptiest-bucket-first protocol described above. Instead of refilling the p emptiest buckets, at each step we refill the buckets sequentially, where, for each of the p refills, we choose the current emptiest bucket and add one unit of water. Thus, the same bucket can be refilled multiple times. We refer to this process of refilling buckets as scheduler $\mathcal{S}_0$. We study $\mathcal{S}_0$ for two reasons: first, understanding this variant will hopefully lead to insights concerning the behaviour of the emptiest-bucket-first scheduler. Second, $\mathcal{S}_0$ also represents an important scenario from a practical perspective. In particular, $\mathcal{S}_0$ corresponds to the case where several processors can service (different threads of) the same job simultaneously. Thus, it can be used when jobs have multiple threads which must all be executed and may be executed simultaneously. We here assume that each job always has at least p available threads.

In this paper, we derive bounds on how much the buckets deviate from the average load A. These bounds translate directly to additive bounds on the deviation between the number of times that a job is serviced in t steps, and the number of times that it should be serviced.

We also prove analogous results for *adversarial systems*, where the sequence k_i is no longer fixed, but instead can vary with time. This corresponds to the situation where the weights of jobs can vary with time, which might occur as a result of changes in the relative importance of the individual jobs, or due to jobs arriving and departing. While this is an important practical consideration, to the best of our knowledge there have not been any previous results proven for proportionate fair scheduling of jobs with varying weights.

We define the system more formally in Section 1.1 and state our results in Section 1.2.

1.1 Models, Terminology and Notation

Let $B_1, \ldots, B_n$ denote a sequence of n buckets, having associated variables $X_1, \ldots, X_n$, where $X_i \in \mathbb{R}$ denotes the amount of material being held in B_i. X_i will be called the *load* of B_i.

The system evolves over discrete time steps, so that $X_i = X_i(t), t = 1, 2, 3, \ldots$. In a single time step, each X_i is first reduced by some amount $k_i \geq 0$. Assume that the total depletion is p, i.e., $p = \sum_j k_j$. We consider algorithms which restore the total load $\sum_j X_j$ by adding the p units back, but are constrained to do so by adding p refills of size 1 to some of the X_i's. Given a rule for selecting which buckets are replenished, we consider how much the X_i's may fluctuate from their original levels.

A system is said to be *stable* whenever there are upper and lower bounds on the values that any of the X_i can take, over time. We are interested in proving stability, and furthermore in identifying bounds on the values of the X_i. The main technical challenge is in finding lower bounds. If at step 1 we have $X_i(1) = A$ for $i = 1, \ldots, n$, we analyse how large A must be to ensure that no bucket ever becomes empty. Let the *intermediate state* $X'_i(t)$ denote the level of the i-th bucket after the depletions in step t but before the refills. Define the *outcome* ψ of the system to be $\inf_{t=1,2,\ldots;\ i=1,\ldots,n} X'_i(t)$, i.e., the greatest lower bound (if it exists) on any bucket load. Thus a stable system is one that has a finite outcome, and we are looking for bounds on the outcome.

FIXED SYSTEMS

In a fixed system, the values k_i are constants, and are the parameters of problem instances. Each k_i is the rate of depletion for B_i. In the t-th step of this basic system, we first *deplete* each $X_i(t)$ by k_i, giving a sequence of *intermediate* values $X'_i(t)$ with $X'_i(t) = X_i(t) - k_i$ for $1 \le i \le n$. Then scheduler $\mathcal{S}_0$ selects some buckets in order to refill them using the following rules. Iteratively for p rounds, $\mathcal{S}_0$ finds an emptiest bucket and adds 1 to its load. This means that one bucket can be refilled more than once in a time step.

ADVERSARIAL SYSTEMS

In *adversarial systems*, we no longer have a fixed sequence of k_i but assume the presence of an adversary. At each step the adversary is free to choose a sequence $k_1(t), \ldots, k_n(t)$ (where t again denotes the time parameter), subject to $k_i(t) \ge 0$, and $\sum k_i(t) = p$. Hence the depletion rate of a bucket can differ from round to round. These systems turn out to be useful to establish bounds on worst-case behaviour.

1.2 Summary of Results

We start with an observation relating the leaky-bucket problem to the performance of the modified SFS algorithm.

OBSERVATION 1. *Consider any fixed system and scheduler. After t steps, the number of times that job i is assigned a processor is* $(tpw_i/\sum_j w_j) + X_i(t+1) - A$.

PROOF. $X_i(t)$ is equal to A plus the number of times that bucket i is refilled minus tk_i. The number of refills is equal to the number of services for job i. Also, recall that $k_i = pw_i/\sum_j w_j$. $\square$

Theorem 1 of Section 2 shows that the outcome ψ of a fixed system with scheduler $\mathcal{S}_0$ is at least $A-(H(n)+p-1)$, where $H(n)$ is the harmonic function, $H(n) = 1+\frac{1}{2}+\frac{1}{3}+\ldots+\frac{1}{n}$. In fact we prove something stronger. The theorem states that (i) There is an adversary $\mathcal{A}_0$ which achieves an outcome of at most $A-(H(n)+p-1)$ against any scheduler $\mathcal{S}$ that refills buckets in p units of size 1 at each step. Also, (ii) Scheduler $\mathcal{S}_0$ achieves an outcome of at least $A-(H(n)+p-1)$ against any adversary $\mathcal{A}$. Thus, scheduler $\mathcal{S}_0$ is optimal in the adversarial setting. Theorem 1 implies that in any fixed system, the outcome ψ is at least $A-(H(n)+p-1)$. This implies that no job lags more than $H(n)+p-1$ steps below the target number of services. In Section 3 we show that Theorem 1 is tight. In particular, Theorem 2 states that for all $\tau > 0$ there are constant depletion rates $k_1, k_2, \ldots, k_n$ such that the minimum load of any bucket is less than $A-(1-\tau)(H(n)+p-1)$ when scheduler $\mathcal{S}_0$ is run. The construction uses a sequence of k_i's that converges to 0 very quickly, suggesting that scheduler $\mathcal{S}_0$ may do better in cases in which the relative values of the k_i's are constrained. Theorem 3 identifies some cases in which this is true. In particular, if each k_j is a multiple of the smallest depletion rate $k_{\min}$ then scheduler $\mathcal{S}_0$ achieves an outcome of at least $A - pk_{\max}/k_{\min}$. Note that for scheduler $\mathcal{S}_0$ there is an easy upper bound of $A+1$ on the maximum load of any bucket. This means that no job gets more than 1 extra service at any point.

2. ADVERSARIAL SYSTEMS

In this section we restrict our attention to adversarial systems, as introduced in Section 1.1. In contrast to fixed systems, we may assume without loss of generality that the states (sequences of X_i values) are sorted in non-decreasing order, i.e., $X(t) = (X_1(t), \cdots, X_n(t))$ with $X_1(t) \le X_2(t) \le \cdots \le X_n(t)$. This means that we "re-arrange" our buckets at the end of every step. We omit the time-parameter t wherever possible (in this case $X = (X_1, \ldots, X_n)$). We use the following definitions. For $1 \le \ell \le n$, $H(\ell) = \sum_{i=1}^{\ell} 1/i$ is the harmonic function, and let

$$\begin{aligned} P_\ell(X) &= \frac{1}{\ell}\sum_{i=1}^{\ell} X_i \\ V_\ell(X) &= P_\ell(X) - H(\ell) \\ V(X) &= \min_\ell V_\ell(X) \\ \ell(X) &= \min\{\ell :\ V_\ell(X) = V(X)\}. \end{aligned}$$

$P_\ell(X)$ is the average of the ℓ lowest buckets; $V_\ell(X)$ measures the outcome achievable by an adversary restricted to $B_1, \ldots, B_\ell$; $V(X)$ is the minimum value of the target function for all prefixes; finally $\ell(X)$ is the length of the shortest prefix that actually takes this minimum.

The following technical lemma is useful later on.

LEMMA 1. *For any state X, $j \in \{1, \ldots, n-1\}$, and any $\alpha > 0$,*

$$\begin{aligned} & P_{j+1}(X) - \alpha \cdot H(j+1) > P_j(X) - \alpha \cdot H(j) \\ \iff\ & X_{j+1} > P_j(X) + \alpha. \end{aligned}$$

PROOF. Consider the following sequence of equivalent inequalities.

$$\begin{aligned} P_{j+1}(X) - \alpha \cdot H(j+1) &> P_j(X) - \alpha \cdot H(j) \\ \frac{1}{j+1}\sum_{k=1}^{j+1} X_k - \alpha \cdot H(j+1) &> \frac{1}{j}\sum_{k=1}^{j} X_k - \alpha \cdot H(j) \\ \frac{X_{j+1}}{j+1} - \frac{\alpha}{j+1} &> \frac{X_1 + \cdots + X_j}{j(j+1)} \\ X_{j+1} - \alpha &> \frac{X_1 + \cdots + X_j}{j} = P_j(X). \end{aligned}$$

$\square$

COROLLARY 1. *For any state X, and $\forall j \in \{1, \ldots, n-1\}$,*

$$V_{j+1}(X) > V_j(X) \iff X_{j+1} > P_j(X) + 1.$$

PROOF. In Lemma 1, let $\alpha = 1$ (recall that $V_j(X) = P_j(X) - H(j)$). $\square$

THEOREM 1. *Let $\mathcal{S}$ be an arbitrary scheduler that refills buckets in p units of size 1 at each step. Let $\mathcal{S}_0$ be the scheduler defined in Section 1.1.*

1. *There is an adversary $\mathcal{A}_0$ which achieves an outcome of at most $A-(H(n)+p-1)$ against any scheduler $\mathcal{S}$.*

2. *The scheduler $\mathcal{S}_0$ achieves an outcome of at least $A-(H(n)+p-1)$ against any adversary $\mathcal{A}$.*

REMARK 1. *Regardless of what scheduler is used, adversary $\mathcal{A}_0$ manages to get the minimum $X_i'(t)$ at least as far as $(H(n)+p-1)$ below the starting line, and irrespective of any strategy used by an adversary, scheduler $\mathcal{S}_0$ will keep the all-time minimum no lower than that level. Hence, in the presence of a "worst-case" mechanism for depleting the buckets, scheduler $\mathcal{S}_0$ is as good as any other scheduler that must refill using p units of equal size.*

PROOF. We put $A = 0$, allow bucket loads to become negative, and establish $-(H(n)+p-1)$ as the bound on how negatively large any X_i need become. Hence, $X(1) = (0,\ldots,0)$.

PROOF OF (1)
To define $\mathcal{A}_0$, we need to define a function that maps any state to a sequence of depletions. Then we show that, if $\mathcal{A}_0$ uses that function to choose depletions, then after finitely many steps the lowest bucket load becomes at most $-(H(n)+p-1)$. Note first that if the minimum load becomes less than or equal to $-(H(n)-1)$ after the scheduler has refilled, then the adversary's choice of depletions is simply to deplete the minimum bucket by p. Next we define the function used by the adversary when the minimum load is greater than $-(H(n)-1)$.

For a state X where the minimum load is no more than $-(H(n)-1)$ (so by the above, the adversary can attain the desired outcome), we have $V(X) \le V_1(X) \le -H(n)$.

Note that for all states X, $V(X) \le V_n(X) = -H(n)$. We have defined $\mathcal{A}_0$ in the case that $\ell(X) = 1$; next we define actions that cause $\ell(X(t))$ to decrease over time.

If $\ell(X) > 1$, $\mathcal{A}_0$'s strategy proceeds in stages. At the start of a stage, suppose $\ell(X) = j > 1$, so $V(X) = P_j(X) - H(j)$. At the end of the stage, some $\bar{X}$ is reached with either $V(\bar{X}) \le V(X) - 1/j$, or with $V(\bar{X}) = V(X)$ and $\ell(\bar{X}) < \ell(X)$. It is sufficient therefore to show that the end of the stage can be reached and satisfies these conditions. This is sufficient because either $V(X(t))$ decreases without limit, or else we recover the case $\ell(X(t)) = 1$ for some t, where $V(X(t)) \le V(X(1))$, and we are done.

$\mathcal{A}_0$'s strategy during the stage is to decrement only X_i for $1 \le i \le j$, reducing one or more to a value which differs from $P_j(X) - 1/j$ by an integer.

If, at any time during this stage, $\mathcal{S}$ refills some X_k, $k > j$, then P_j is reduced by at least $1/j$, reaching a state $\bar{X}$ where $V(\bar{X}) \le V_j(\bar{X}) \le P_j(X) - 1/j - H(j) \le V(X) - 1/j$.

Otherwise, since $\mathcal{S}$ can only refill in integer quantities, this strategy must eventually succeed in reaching a state $\bar{X}$ where $P_j(\bar{X}) = P_j(X)$ and each of $\bar{X}_1,\ldots,\bar{X}_j$ differs from $P_j(X) - 1/j$ by an integer. (To see this, note that the adversary is trying to remove some fractional amount from each X_i and the scheduler may only replenish buckets in integer amounts.) The largest of these quantities, $\bar{X}_j$, must satisfy $\bar{X}_j \ge P_j(X) + 1 - 1/j$. Hence

$$\begin{aligned}
V(\bar{X}) &\le V_{j-1}(\bar{X}) = \frac{(\bar{X}_1 + \cdots + \bar{X}_j) - \bar{X}_j}{j-1} - H(j-1) \\
&\le \frac{jP_j(X) - P_j(X) + (\frac{1}{j} - 1)}{j-1} - H(j-1) \\
&= P_j(X) - H(j) = V(X),
\end{aligned}$$

and so $\ell(\bar{X}) < \ell(X)$.

PROOF OF (2)
Suppose for a contradiction that there exists an adversary $\mathcal{A}$ which achieves an outcome of less than $-(H(n)+p-1)$ against $\mathcal{S}_0$. This means that after t time steps for some finite t, state $X(t)$ satisfies $X_1(t) < -(H(n)-1)$ (this is because $\mathcal{A}$ would deplete the smallest bucket by p in order to minimise the outcome), i.e. $P_1(X(t)) < -(H(n)-1)$, hence $V_1(X(t)) < -H(n)$. Observe that, by definition, for all states X, $V_n(X) = -H(n)$ since the overall average is always 0. Initially, $V_j(X(1)) > V_{j+1}(X(1))$ for all $j < n$ since we assume all buckets to start with load 0, i.e., the harmonic numbers are all that count. Finally, $V(X(t)) < -H(n) = V_n(X(t))$.

Consider the first state $\bar{X}$ at which $V_j(\bar{X}) < -H(n)$ for some j. From our observation above we have $V_n(\bar{X}) = -H(n)$, so we can choose the smallest j so that we also have $V_j(\bar{X}) < V_{j+1}(\bar{X})$. Suppose X^{prev} is the previous state, and we will show that $V_j(X^{\text{prev}}) < -H(n)$, contradicting our assumption that $\bar{X}$ was the first state with the given property (which is not a property of $X(1)$). Since $V_j(\bar{X}) < V_{j+1}(\bar{X})$ we have by Corollary 1

$$\bar{X}_{j+1} > P_j(\bar{X}) + 1 = \frac{\bar{X}_1 + \cdots + \bar{X}_j}{j} + 1.$$

Now observe that $\mathcal{S}_0$ cannot have just refilled any $\bar{X}_k$ for $k > j$, just prior to reaching state $\bar{X}$. Since those buckets have a load more than 1 above the average of the lowest j buckets, it follows that the lowest bucket has a load less than $\bar{X}_k - 1$ for $k > j$. $\mathcal{S}_0$ would certainly have preferred one of $\bar{X}_1,\ldots,\bar{X}_j$, which would be lower prior to a refill of size 1. If $Y_1,\ldots,Y_j$ denote the loads of the buckets corresponding to $\bar{X}_1,\ldots,\bar{X}_j$ in state X^{prev} (these are not necessarily $X_1^{\text{prev}},\ldots,X_j^{\text{prev}}$), then $Y_1 + \cdots + Y_j \le \bar{X}_1 + \cdots + \bar{X}_j$, since $\mathcal{S}_0$ added p to these buckets and $\mathcal{A}$ previously subtracted at most p. Hence

$$P_j(X^{\text{prev}}) \le \frac{Y_1 + \cdots + Y_j}{j} \le P_j(\bar{X}).$$

So $V_j(X^{\text{prev}}) \le V_j(\bar{X}) < -H(n)$, contradicting the choice of $\bar{X}$. $\square$

3. FIXED SYSTEMS

In Section 2 we showed that scheduler $\mathcal{S}_0$ keeps the all-time minimum to $A - (H(n) + p - 1)$ against an adversary that may select the depletion rates in every step. Of course, this upper bound on the initial bucket load holds also in the case of our fixed system. This gives the following corollary.

Corollary 2. *If $A \geq H(n)+p-1$, then loads $X_i(t)$ will always be non-negative, for any leakage rates $k_1, \ldots, k_n$.*

The next theorem presents a construction of leakage rates that show that the above condition $A \geq H(n)+p-1$ is necessary as well as sufficient. However it uses a very large ratio between highest and lowest depletion rates. This motivates the restriction to rational depletion rates, where bounds are obtained in terms of the above ratio.

In Theorem 3 we will show an alternative upper bound on the initial bucket load A needed to maintain positive loads. This gives better results in the case that the k_i's are all small multiples of a common value.

3.1 Arbitrary Real-valued Depletion Rates

In the following we show that in the worst case (worst case over all choices of k_i's summing to p) a bucket may be depleted by up to $H(n) + p - 1$. An alternative statement is that, assuming all buckets are initially set to the same level, they must start with a level of $A = H(n) + p - 1$ in order to avoid becoming empty.

The worst-case behaviour we obtain for constant depletion rates essentially matches the upper bound on worst-case behaviour for the adversarial case. More precisely, we show how to construct sets of depletion rates that lead to a minimum bucket value that is arbitrarily close to the $A-(H(n)+p-1)$ obtained in the adversarial case.

Theorem 2. *For all $\tau > 0$ there exist constant depletion rates $k_1, k_2, \ldots, k_n$ such that the minimum load of any bucket is less than $A - (1-\tau)(H(n) + p - 1)$.*

In the following we assume $A = 0$ and ask how negatively large can the value of any bucket become. Thus we show that some bucket may be depleted to a level arbitrarily close to $-(H(n) + p - 1)$, for suitable choice of the k_i.

The general idea of the proof is to have $k_1 \gg k_2 \gg \ldots \gg k_n$. We use an inductive argument that for each i, the first i buckets $B_1, \ldots, B_i$ attain a lowest level of about $-(H(i) + p-1)$ (assuming all start at 0) in time inversely proportional to k_i. Moreover, there is a cyclic behaviour which guarantees that the loads $X_1, \ldots, X_i$ of the first i buckets repeatedly return to become approximately equal, after a number of steps inversely proportional to k_i.

Definition 1. *Define an n-*bucket surplus system *to be a set of buckets $B_1, \ldots, B_n$ with associated depletion rates $k_1, \ldots, k_n$ which sum to a quantity* **at most** *p. Define the* surplus *of an n-bucket surplus system with depletion rates $k_1, \ldots, k_n$ to be $p - (k_1 + \ldots + k_n)$.*

Thus, over time the total load of an n-bucket surplus system increases whenever the total depletion rate is strictly less than p, and in particular, increases by the surplus at each step.

Notation: Let $\psi_p(k_1, \ldots, k_i)$ denote the lowest level attainable by any bucket in the i-bucket surplus system that has those depletion rates. Let $\psi_p(i)$ be $\inf_{(k_1,\ldots,k_i)} \psi_p(k_1, \ldots, k_i)$.

Theorem 2 states that $\psi_p(n) = -(H(n) + p - 1)$.

Definition 2. *Let S be an i-bucket surplus system. For $r = 0, \ldots, p-1$, an r-*th intermediate state *is the sequence of bucket loads obtained after the first r refills. Let μ be the smallest depletion rate of any bucket in S. We say that S has* tolerance *τ if the following holds. In every sequence of $4p/\mu$ steps:*

1. *there is a step when all buckets are within τ of their average;*
2. *there is a step when buckets $\mathcal{B}_1, \ldots, \mathcal{B}_{i-1}$ are within τ of their own average, and bucket $\mathcal{B}_i$ (assumed to have the smallest depletion rate) is at least $1-\tau$ above the average of the others;*
3. *there is a intermediate state in which the smallest bucket load is at least $(1-\tau)(H(i) + p - 1)$ below the average value.*

A system with small tolerance is one that is close to achieving the claimed worst-case behaviour (and also achieves it repeatedly, over a cycle whose length is inversely proportional to the smallest depletion rate in the system). Tolerance is a measure of how much a system may fall short of being worst-case. We conjecture that there is no system with zero tolerance; here we show how to construct systems with arbitrarily small tolerance.

Lemma 2. *Suppose that $\delta > \delta' \geq 0$. Any i-bucket surplus system S with surplus δ can be converted into an i-bucket surplus system S' with surplus δ', and $\psi_p(S') \leq \psi_p(S)$. If S has tolerance τ then S' has tolerance τ.*

Proof. To obtain S' from S, just increase the depletion rates of buckets in S by $(\delta - \delta')/i$. Then S' behaves like S (for all t, at step t the same buckets are refilled by the same amounts), the only difference being that $X_i(t)$ is lower by $t(\delta - \delta')/i$, for all buckets B_i and all t. (In cases where two or more buckets may be joint minimal, any sequence of tie-breaks for S has a corresponding sequence of tie-breaks for S', and vice versa.)

The relative values of bucket loads are unchanged by the operation (they are all shifted down by the same amount), and tolerance is a measure of relative values of buckets (in particular the differences between loads). □

Lemma 3. *For all positive integers i and $\tau > 0$, there is an i-bucket surplus system S with some surplus $\delta > 0$ having tolerance τ (using scheduler $\mathcal{S}_0$ to refill buckets).*

For our purposes the relevant fact that follows from the tolerance of S is the lower bound attained on the difference between average and minimum levels. The additional information about the behaviour of the system is needed for the inductive step. The positive surplus can be set to zero by using Lemma 2 (with $\delta' = 0$) to obtain a standard zero-surplus system where the average load is always 0, and hence the minimum level reaches $-(1-\tau)(H(i)+p-1)$.

PROOF. We proceed by induction on i, the number of buckets in a system. Suppose we have some value τ for which we want to construct an i-bucket system with tolerance τ. We may assume in the following that τ is a small quantity, since a successful construction with a small tolerance constitutes a successful construction with larger tolerance.

We can assume inductively that we have an $(i-1)$-bucket surplus system S' with tolerance $\tau/10p$, and surplus $\delta' > 0$. Let $k_1, \ldots, k_{i-1}$ be the depletion rates of the buckets in S', with $k_1 \geq k_2 \geq \ldots \geq k_{i-1}$.

We construct an i-bucket system S as follows.

- The first $i-1$ buckets are given depletion rates $k_1, k_2, \ldots, k_{i-1}$.
- Put $k_i = \min\{\delta'/2, \tau k_{i-1}/20p\}$.
- Re-scale the surplus δ of the resulting system to be at most $\tau k_i/9ip$, by using the method of Lemma 2 applied to $k_1, \ldots, k_{i-1}$ but not k_i. (So $k_1, \ldots, k_{i-1}$ increase slightly.)

We argue that S constructed in this way has tolerance τ and positive surplus. Note that in the expression $\min\{\delta'/2, \tau k_{i-1}/20p\}$, the first argument $\delta'/2$ ensures that we do not use up more than half the remaining surplus available, since the total depletion rate is supposed to be less than p. The third step ensures that the surplus is small enough that by the time we observe a large difference between the average and smallest bucket values, we still have $\mathrm{avg_j(X_j)}$ (the average of the X_j's) very small.

The initial state of S satisfies Condition 1 of the definition of tolerance. We first show that if we start at any state where the values satisfy Condition 1, then after about $\frac{1}{k_i}$ steps, we fulfill Condition 2 of the claimed tolerance τ. Thus, all X_j are approximately equal at a $(p-1)$-st intermediate state, and then B_i is refilled. After that, we verify that the system satisfies Condition 3 after fewer than $\frac{1}{k_i}$ further steps. Then we show that after about $\frac{1}{k_i}$ steps the system returns to a state satisfying Condition 1 where all X_j are approximately equal.

B_i has the slowest depletion rate, and we consider the sequence of steps prior to B_i being refilled for the first time. In each step, X_i decreases by k_i and $\mathrm{avg_j(X_j)}$ increases by δ/i where δ is the surplus. For system S with surplus δ, at an r-th intermediate state Y after t steps,

$$\begin{aligned} & -tk_i - \frac{\delta t - (p-r)}{i} + \tau \\ \geq\ & Y_i - \mathrm{avg_j(Y_j)} \\ \geq\ & -tk_i - \frac{\delta t - (p-r)}{i} - \tau. \end{aligned}$$

For B_i to be refilled here we have as a necessary condition, $Y_i - \mathrm{avg_j(Y_j)} \leq 0$, and from the second of the above pair of inequalities we can infer the necessary condition

$$t \geq \Big(\frac{p-r}{i} - \tau\Big) / \Big(k_i + \frac{\delta}{i}\Big).$$

For $r < p-1$, we would need

$$t \quad \geq \quad \frac{\frac{2}{i} - \tau}{k_i + \frac{\delta}{i}} > \frac{\frac{2}{i}}{k_i + \frac{k_i}{9}} > \frac{8}{5ik_i},$$

since we have chosen $\delta \leq \tau k_i/9ip < ik_i/9$ and τ sufficiently small.

We now show that in fact B_i is refilled when $r = p-1$, by using the inductive hypothesis to establish a sufficient condition for B_i to be refilled.

Put $r = p-1$ and then $t > (\frac{\tau}{10p} + \frac{1}{i} + \tau)/(k_i + \frac{\delta}{i})$ is a sufficient condition to have $Y_i - \mathrm{avg_j(Y_j)} < -\tau/\mathrm{10p}$, at the $(p-1)$-st intermediate step. By the inductive hypothesis, the elements of S' (i.e., $B_1, \ldots, B_{i-1}$) are within $\frac{\tau}{10p}$ of their average at a $(p-1)$-st intermediate state, during every $4/k_{i-1}$ steps. Suppose X_i becomes as low as $\tau/10p$ below $\mathrm{avg_j(X_j)}$. Within the next $4/k_{i-1}$ steps we reach a $(p-1)$-st intermediate state Y where

- $Y_i - \mathrm{avg_{j=1\ldots,i-1}(Y_j)} < \frac{-\tau}{\mathrm{10p}}$, and
- for all $j < i$, Y_j is within $\tau/10p$ of $\mathrm{avg_{k=1,\ldots,i-1}(Y_k)}$ $\geq \mathrm{avg_{k=1,\ldots,i}(Y_k)}$.

Hence Y_i is lowest, so B_i is refilled. If $t = \lceil(\frac{\tau}{10p} + \frac{1}{i} + \tau)/k_i + \frac{4}{k_{i-1}}\rceil$ then $t \leq \frac{11}{10ik_i} + \frac{4}{k_{i-1}}$, and we encounter a sufficient condition for B_i to be refilled at a $(p-1)$-st intermediate step before we encounter a necessary condition for B_i to be refilled at an r-th intermediate step, for any $r < p-1$. During the sequence of $4/k_{i-1}$ steps, $X_i - \mathrm{avg_{j=1,\ldots,i-1}(X_j)}$ decreased by $\frac{4}{k_{i-1}} \cdot (k_i + \frac{\delta}{i}) \leq \frac{2\tau}{10p}$ using expressions we chose for δ and k_i in terms of k_{i-1}. We conclude that when B_i is refilled for the first time,

$$X_i - \mathrm{avg_{j=1\ldots i-1}(X_j)} \geq 1 - \frac{3\tau}{\mathrm{10p}}.$$

Observe also that at this point we fulfill Condition 2 for S to have tolerance τ, because all other buckets are within τ of their own average and X_i is more than $1-\tau$ above. We next argue that we do not have to wait much further until Condition 3 is fulfilled (the large gap between smallest bucket and the average).

After B_i is refilled for the first time, the inductive hypothesis states that within an additional $4p/k_{i-1}$ steps,

$$\begin{aligned}&\mathrm{avg_{j=1,\ldots,i-1}(X_j)} - \mathrm{min\{X_1,\ldots,X_{i-1}\}}\\ \geq\ &(1-\frac{\tau}{10p})(H(i-1)+p-1)\end{aligned}$$

The general idea now is to show a lower bound of approximately $\frac{1}{i}$ on $\mathrm{avg_{j=1,\ldots,i}(X_j)} - \mathrm{avg_{j=1,\ldots,i-1}(X_j)}$, and derive a difference of at least $(1-\tau)(H(i)+p-1)$ between $\min_j(X_j)$ and $\mathrm{avg_{j=1,\ldots,i}(X_j)}$.

After the at most $4/k_{i-1}$ steps that it takes to observe the difference of $(1-\frac{\tau}{10p})(H(i-1)+p-1)$ noted above, X_i is still too far above average for B_i to be refilled again, and we have

$$X_i - \mathrm{avg_j(X_j)} \geq 1 - \frac{3\tau}{10\mathrm{p}} - \frac{4}{\mathrm{k_{i-1}}}\Big(\mathrm{k_i} + \frac{\tau \mathrm{k_i}}{9\mathrm{i}^2\mathrm{p}}\Big)$$

$$\Longrightarrow \quad X_i - \mathrm{avg_j(X_j)} \geq 1 - \frac{\tau}{2\mathrm{p}}$$

using the expression we chose for k_i in terms of k_{i-1}. Furthermore, at the point when one of the first $i-1$ buckets falls short of their own average by $(1-\frac{\tau}{10p})(H(i-1)+p-1)$ we have:

$$\begin{aligned}&\mathrm{avg_{j=1\ldots i}(X_j)} - \mathrm{avg_{j=1\ldots i-1}(X_j)}\\ \geq\ &\frac{1}{i}\Big(\text{lower bound on } (X_i - \mathrm{avg_{j=1\ldots i-1}(X_j)})\Big)\\ =\ &\frac{1}{i}\Big(1-\frac{\tau}{2p}\Big)\end{aligned}$$

Hence some bucket falls short of the overall average by at least

$$\Big(1-\frac{\tau}{10p}\Big)\Big(H(i-1)+p-1\Big)+\frac{1}{i}\Big(1-\frac{\tau}{2p}\Big) \geq (1-\tau)(H(i)+p-1)$$

as required.

We have shown how, starting from an initial state where all values are approximately equal (such as the standard all-zeros initial state), we find states satisfying Condition 2 then Condition 3. The argument does not require the average to be zero, it can be any real number (as needed in subsequent cycles of the system, where the average will have increased due to the surplus). Note however that we need to use a surplus that is small in comparison with the depletion rates.

Finally, to see that the system also returns to a state satisfying the first condition, the idea is to wait until $X_i - \mathrm{avg_{j=1,\ldots,i}(X_j)}$ is in the range $[\tau-\frac{1}{i}, \tau/2-\frac{1}{i}]$. When $X_i - \mathrm{avg_{j=1,\ldots,i}(X_j)}$ is in this range, X_i cannot be refilled, but is less than τ above average at a $(p-1)$-st intermediate step. While $X_i - \mathrm{avg_{j=1,\ldots,i}(X_j)}$ is in this range, at least $(\tau/2)/k_i$ steps elapse. Noting that $(\tau/2)/k_i > 4/k_{i-1}$, the inductive hypothesis lets us claim that, during this time, buckets $B_1,\ldots,B_{i-1}$ reach a $(p-1)$-st intermediate state that satisfies the condition with tolerance $\tau/10p$, and, at that point, all the buckets $B_1,\ldots,B_i$ are in an intermediate state that satisfies the first condition with tolerance τ. ☐

Theorem 2 follows from Lemma 3, using the tolerance τ in Lemma 3 as the τ in the statement of Theorem 2.

3.2 Rational Depletion Rates

In this section we will show that the system behaves much more nicely if the depletion rates are "nice", for example if they are multiples of each other. To state the results we will assume in the following that all the k_i's are positive rationals. As before we have $k_1 + \ldots + k_n = p$, but now we assume, without loss of generality, that $k_j = pm_j/M$ for $1 \leq j \leq n$, where $M = \sum_j m_j$ and the greatest common divisor of all of the m_j's is 1. All the results so far stated in this paper also hold in this setting.

For all $t \geq 0$ we have $\sum_j X_j(t) = nA$ and $\sum_j X'_j(t) = nA - p$. Furthermore, note that no bucket before its refill ever has a load larger than $A - 1/n$, hence $A - 1/n + 1$ is an upper bound on the maximum bucket load.

In the following we will show that the system is periodic. This result will be used below to give a lower bound for the outcome achieved by scheduler $\mathcal{S}_0$.

Lemma 4. *For rational values, $k_j = pm_j/M$, $1 \leq j \leq n$, the system returns to its initial state after M steps.*

Proof. During this period of M steps, any bucket B_i is refilled at most pm_i times, because otherwise it would have a load of at least $A - k_iM + (pm_i+1) = A+1$. This contradicts the observation above that no bucket will ever have a load exceeding $A - 1/n + 1 < A + 1$. Since $\sum_j(pm_j) = pM$, each bucket B_i must be filled *exactly* $pm_i = k_iM$ times, and so the system returns to its initial state. ☐

Theorem 3. *Let $k_1 \geq k_2 \geq \ldots \geq k_n$ be the set of depletion rates, where $k_j = pm_j/M$, for $1 \leq j \leq n$, the gcd of the (integer) m_j's is 1, and $\sum_j m_j = M$.*

1. *Scheduler $\mathcal{S}_0$ achieves an outcome of at least $A - pm_1$.*
2. *If each k_j is a multiple of k_n, then scheduler $\mathcal{S}_0$ achieves an outcome of at least $A - pk_1/k_n$.*

Proof. According to Lemma 4, the system returns to its initial state after M steps, and so also at any multiple of M steps. Hence, the deviation of B_i's load from the initial value is bounded above by $k_iM \leq k_1M = pm_1$.

If each k_j is a multiple of k_n then we have $m_n = 1$ and the result follows, since $m_1/m_n = k_1/k_n$. ☐

4. CONCLUSIONS AND FURTHER WORK

We think the major open question is to prove similar results for the non-threaded version of SFS where we simply increase the p smallest buckets by one, without allowing one bucket to be increased twice per step. Unfortunately, this scheduling rule is much harder to analyze. The main problem seems to be that the nice property of S_0 does not hold any longer: no bucket that is refilled ever has a load larger than $A - 1/n$.

Our bounds on deviation from fairness do not depend on t (the elapsed number of time quanta), and it may be that the largest deviation possible grows very slowly as a function of t, and that as a result better bounds could be found in terms of t. Note that quanta are typically on the order of milliseconds (sometimes even smaller), and jobs can run on the order of days. Hence t might be somewhere around 10^9, for example. For this range of t we may actually be able to get better worst-case behaviour; at least it is not ruled out by our construction of Theorem 2. An alternative approach is to obtain better bounds in terms of the ratio between largest and smallest values of the depletion rates, which in Theorem 2 is more than exponential in n.

We believe that the result for the adversarial model in which the k_i's may be chosen by an adversary, should be adaptable to a situation where the number of buckets is allowed to vary. Given an upper bound N on the number of buckets that may be present, then for a step with $n < N$ buckets present, we could have $N - n$ buckets with $k_i = 0$. It seems likely that the results showing the buckets do not fall too far below average could be made to apply to this case. Note that the result is not immediate, since the buckets with $k_i = 0$ could be refilled.

5. REFERENCES

[1] J. Anderson and A. Srinivasan. A new look at pfair priorities. Technical Report TR00-023, Dept of Computer Science, Univ. of North Carolina, 2000.

[2] J. Anderson and A. Srinivasan. Early-release fair scheduling. In *Proceedings of the 12th Euromicro Conference on Real-Time Systems, Stockholm, Sweden*, pages 35–43, 2000.

[3] S. K. Baruah, N. K. Cohen, C. G. Plaxton, and D. A. Varvel. Proportionate progress: A notion of fairness in resource allocation. *Algorithmica*, 15:600–625, 1996.

[4] S. Baruah, J. Gehrke, and C. G. Plaxton. Fast scheduling of periodic tasks on multiple resources. In *Proceedings of the Ninth International Parallel Processing Symposium, Santa Barbara, CA*, pages 280–288, 1996.

[5] A. Chandra, M. Adler, P. Goyal, and P. Shenoy. Surplus fair scheduling: A proportional-share cpu scheduling algorithm for symmetric multiprocessors. In *Proceedings of the Fourth Symposium on Operating System Design and Implementation (OSDI 2000), San Diego, CA*, pages 45–58, 2000.

[6] A. Chandra, M. Adler, and P. Shenoy. Deadline fair scheduling: Bridging the theory and practice of proportionate-fair scheduling in multiprocessor servers. In *Proceedings of the Seventh IEEE Real-time Technology and Applications Symposium (RTAS 2001), Taipei, Taiwan*, pages 3–14, 2001.

[7] K. Duda and D. Cheriton. Borrowed virtual time (bvt) scheduling: Supporting latency-sensitive threads in a general-purpose scheduler. In *Proceedings of the 17th ACM Symposium on Operating Systems Principles (SOSP'99), Kiawah Island Resort, SC*, pages 261–276, 1999.

[8] S.J. Golestani. A self-clocked fair queueing scheme for broadband applications. In *Proceedings of IEEE INFOCOM 1994, Toronto, Canada*, pages 636–646, 1994.

[9] P. Goyal, X. Guo, and H.M. Vin. A hierarchical cpu scheduler for multimedia operating systems. In *Proceedings of the Second Symposium on Operating System Design and Implementation (OSDI'96), Seattle, WA*, pages 107–121, 1996.

[10] M. Moir and S Ramamurthy. Pfair scheduling of fixed and migrating periodic tasks on multiple resources. In *Proceedings of the 20th Annual IEEE Real-Time Systems Symposium, Phoenix, AZ*, pages 294–303, 1999.

[11] J. Nieh and M S. Lam. The design, implementation and evaluation of smart: A scheduler for multimedia applications. In *Proceedings of the 16th ACM Symposium on Operating Systems Principles (SOSP'97), Saint-Malo, France*, pages 184–197, 1997.

[12] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks - The single node case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, 1993.

[13] C. A. Waldspurger and W. E. Weihl Stride scheduling: Deterministic proportional-share resource management. Technical report MIT/LCS/TM-528, 1995.