A Maiden Analysis of Longest Wait First

Jeff Edmonds* Kirk Pruhs†

Abstract

We consider server scheduling strategies to minimize average flow time in a multicast pull system where data items have uniform size. The algorithm Longest Wait First (LWF) always services the page where the aggregate waiting times of the outstanding requests for that page is maximized. We provide the first non-trivial analysis of the worst case performance of LWF. On the negative side, we show that LWF is not s-speed $O(1)$-competitive for $s < \frac{1+\sqrt{5}}{2}$. On the positive side, we show that LWF is 6-speed $O(1)$-competitive.

1 Introduction

In a pull-based client-server system, the server receives client initiated requests for items/pages/documents over time. In a multicast/broadcast system, when the server sends a requested page, all outstanding client requests to this data item are satisfied by this multicast. The system may use broadcast because the underlying physical network provides broadcast as the basic form of communication, for example if the network is wireless or the whole system is on a LAN. Multicast may also arise in a wired network as a method to provide scalable data dissemination. See for example multicast based scalable data dissemination systems described in [6, 7]. One commercial example of a multicast-pull client-server system is Hughes' DirecPC system [8]. In the DirecPC system the clients request documents via a low bandwidth dial-up connection, and the documents are broadcast via high bandwidth satellite to all clients.

In this paper we consider a setting where the date items, or pages, are of approximately the same size. This might arise, for example, if the server is a DNS server. We consider the objective function of minimizing the average flow/response/waiting time of the client requests. The preponderance of evidence to date indicates that the "right" algorithm for this problem is Longest Wait First (LWF). LWF always services the page where the aggregate waiting times of the outstanding requests for that page is maximized. All the experimental comparisons of the most natural algorithms have identified the LWF as the clear champion [9, 23, 2]. In the natural setting where the request arrival times for each page have a Poisson

*York University, Canada. jeff@cs.yorku.ca. Supported in part by NSERC Canada.

†Computer Science Department. University of Pittsburgh. kirk@cs.pitt.edu. Supported in part by NSF grant CCR-0098752, NSF grant ANI-0123705, and NSF grant CNS-0325353.

distribution, LWF broadcasts each page with frequency roughly proportional to the square root of the page's arrival rate, which is essentially optimal [3].

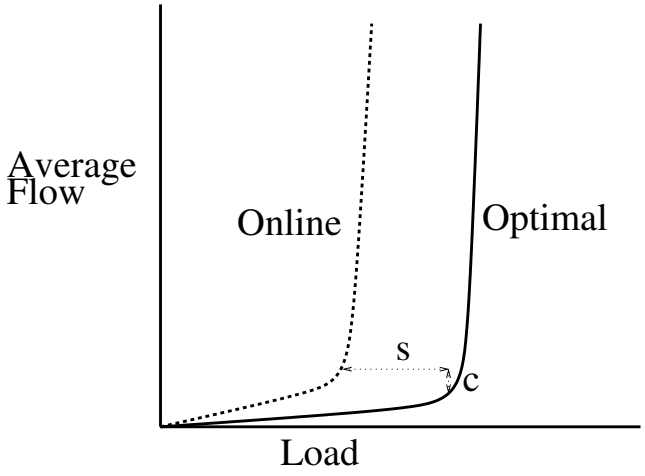

Figure 1: The worst possible performance curve of an s-speed c-competitive online algorithm.

To understand the worst-case analysis results in the literature, we need to introduce and motivate resource augmentation analysis. Resource augmentation analysis was proposed as a method for analyzing scheduling algorithms in [16]. We adopt the notation and terminology from [20]. In the context of our problem, an s-speed c-competitive algorithm A has the property that $\max_I \frac{A_s(I)}{\mathrm{OPT}_1(I)} \leq c$ where $A_s(I)$ denotes the average flow time for the schedule that algorithm A with a speed s processor on input I, and similarly $\mathrm{OPT}_1(I)$ denotes the flow time of the adversarial schedule for I with a unit speed processor. Our analysis philosophy is to put first priority on minimizing the speed, while keeping the competitive ratio reasonable, ideally $O(1)$. The reason for this is average quality of service (QoS) curves such as those in figure 1 are ubiquitous in server systems. That is, the average QoS at loads below capacity is negligible, and the average QoS above capacity is intolerable. The concept of load is not so easy to formally define, but generally reflects the rate at which work arrives at the server. So in some sense, one can specify the performance of such a system by simply giving the value of the capacity of the system. In this setting, $A_s(I)$ is at most c times optimal average flow time with s times higher load, since slowing down the speed by a factor of s is the same as increasing the load by a factor of s. But since the optimal flow time is almost always negligible or intolerable, a modest c times either negligible and intolerable, still gives you negligible or intolerable. So an s-speed c-competitive algorithm should perform reasonably well up to load $1/s$ of the capacity of the system as long as c is of modest size. An algorithm that is $(1+\epsilon)$-speed $O(1)$-competitive is said to be *almost fully scalable* [21].

Worst-case analysis of this multicast pull scheduling problem was initiated in [17]. They showed that there is no O(1)-competitive online algorithm for this problem. They also showed the intuitive greedy algorithm, Most Requests First (MRF), is not even $O(1)$-speed

$O(1)$-competitive. MRF always services the page with the most outstanding requests. This result perhaps explains the experimental inferiority of MRF observed in [9, 23, 2]. In [17] it was observed that $O(1)$-competitiveness for $O(1)$-speed online algorithms can not be proved using local competitiveness[21]. Building on the work in [10], in [11] it was shown that there exists an algorithm EDF-EQUI that is $(4+\epsilon)$-speed $O(1+1/\epsilon)$-competitive. Unfortunately, BEQUI-EDF is not a particularly natural algorithm, and it is known that BEQUI-EDF can not be 2-speed $O(1)$-competitive.

1.1 Our Results In [17] is was conjectured that LWF was almost fully scalable. We refute this conjecture in section 3 by showing that that LWF is not s-speed $O(1)$-competitive for $s < \frac{1+\sqrt{5}}{2}$. On the positive side, we show in section 2 that LWF is 6-speed $O(1)$-competitive. Intuitively, this result should probably be viewed as saying that LWF's performance is at least reasonable. This is the first non-trivial analysis of the worst case performance of LWF.

1.2 Related Results In order to obtain a positive result for multicast pull scheduling, [17] resorted to considering offline algorithms. They found an LP-based polynomial-time algorithm that is 3-speed 3-competitive. Subsequently better results were obtained by using different rounding schemes. In [12] a rounding that is 6-speed 1-competitive algorithm is given. In [13] a rounding that is 2-speed 2-competitive is given. In [14] a rounding that is 3-speed 1-competitive is given. Finally, in [4], a rounding that is $O(1+\epsilon)$-speed $O(1)$-competitive is given. Note that all of these algorithms are offline and thus are not implementable in a server. This problem was proven to be NP-hard in [12].

There has also been some worst case analysis of multicast pull scheduling algorithms in the case that data items have different sizes and where preemption is allowed. This would be a more appropriate formalization of the DirecPC system for example. Firstly there is more than one reasonable way to formalize this problem, for example, the clients may or may not be required to receive the requested file in order. In [22] different alternative formalizations of the problem are compared. Building on the work of [10], it is shown in [11] that the algorithm B-Equi, which broadcasts each document at a rate proportional to the number of outstanding requests for that document, is $(4+\epsilon)$-speed $O(1+1/\epsilon)$-competitive in all the reasonable models. In [5] some results for maximum flow time are given. Experimental results in the case of arbitrary data item size can be found in [1, 15]. LWF can be implemented in logarithmic time per query [18]. For a survey on online scheduling, including multicast pull scheduling, see [21]. Problems that are special cases of multicast pull scheduling included weighted flow time, and scheduling jobs with sequential/parallel speed-up curves [11, 21].

There has been a fair amount of research into push-based broadcast systems, sometimes called broadcast disks, where the server pushes information to the clients without any concept of a request, ala a television or radio broadcast. See for example [19]. A survey of the literature of both pull and push multicast/broadcast scheduling can be found in [24].

1.3 Definitions and Preliminaries We formalize the problem in the following way. The setting consists of a corpus $\mathcal{P}$ of n possible pages. We assume that time is discrete. Suppose we have a s-speed server. Consider some time t. First the server receives $R_i(t) \geq 0$ new requests for page $i \in \mathcal{P}$ (for each $1 \leq i \leq m$). The s-speed server then decides on up to s pages to broadcast at time t. We say that the $R_i(t)$ requests for page i at time t are *satisfied* at time $C_i(t)$ if $C_i(t) \geq t$ is the first when P_i is broadcasted after these requests arrived. In the online setting, the server is not aware of the requests $R_i(t)$ until time t.

We consider the objective function of minimizing the total (or equivalently average) system performance. The *response/flow* time of a request for page $i \in \mathcal{P}$ at time t is $(C_i(t) - t + 1)$, which is how long a client that requested page i at time t has to wait until page i is broadcasted. Note that the plus 1 term is to assure that we don't get flow times of 0. The *total response time* is then $\sum_t \sum_i R_i(t) \cdot (C_i(t) - t + 1)$.

The current *wait* of a page is defined as follows. At time t, let $\hat{C}_i(t)$ be the last time, strictly before time t, that LWF broadcast page i. If there is no such time, then $\hat{C}_i(t) = 0$. Then the wait of a page i at time t is $\sum_{\hat{C}_i(t) < t' \leq t} R_i(t') \cdot (t - t' + 1)$. At each time t, LWF broadcasts the s pages with largest wait. It will be convenient in out proofs to think of LWF ordering these s broadcasts by decreasing wait. So the first of the s broadcasts at time t, is the page with the highest total wait.

As is common when analyzing scheduling algorithms, we need a time/event ordering not only for the times in the schedule, but for also for the events internal to the scheduling algorithms. Assume that each of S and T is either an event in the execution of LWF, or a time. We then use the notation $S \lesssim T$ to mean that S happened before T in LWF's time. If for example, S is the broadcast of a page i at time t and T is the broadcast of a page j at time t then $S \lesssim T$ is equivalent to saying the i's wait is greater than j's wait at time t.

The time between the ith and jth, $i < j$, broadcasts by LWF might be either $\lfloor \frac{j-i}{s} \rfloor$ or $\lceil \frac{j-i}{s} \rceil$. To avoid ugly floors and ceilings in our analysis, we estimate the length of such a time period by $(j - i)/s$. This additive error will not affect our flow time estimations by more than a multiplicative constant.

2 Analysis of LWF

Our goal in the section is to prove the following theorem:

THEOREM 2.1. *For all instances I,* $\frac{\mathrm{LWF}_6(I)}{\mathrm{OPT}_1(I)} = O(1)$

We benevolently assume that when LWF services a page, all outstanding requests for this are *implicitly* serviced for the adversary. This allows us to simplify our arguments in two ways. Firstly, because all requests for this page are serviced by both schedulers, we can analyze subsequent requests to this page independently. Secondly, since this assumption implies that the adversary never falls behind LWF on any page, this greatly reduces the number of cases that we have to consider. We further assume, without loss of generality,

that LWF broadcasts s pages on every step. Otherwise, we can just apply our argument to every maximal time interval where this holds. From here on we fix a particular instance I and drop it from the notation.

In this paragraph we give an informal road map of our analysis. We develop an accounting scheme in which the adversary's costs eventually pay for all of LWF's costs. The currency of payment is not actually wait time, which depends on the number of requests, but is the number of time steps in which servicing is delayed. The proof first identifies *A-costly-events* which are more costly for the adversary because LWF services them quickly and *L-costly-events* that are more costly for LWF because it services them long after the adversary does. The proof transforms the original input into a canonical input to make this distinction even more pronounced. LWF is initially delayed by A-costly-events. These A-costly-events pay for the L-costly-events that they delay. L-costly-events are payed more than they actually need so that they can in turn pay for the L-costly-events that they delay. When an event is in the role of paying, it will either be referred to as an *A-paying-event* or as an *L-paying-event* depending on whether it is A or L-costly. Similarly, when an L-costly-event is in the role of being payed, it will be referred to either as a *payed-by-A-event* or as a *payed-by-L-event*. An L-costly-event begins being costly when the adversary services it. The events that LWF is servicing during this same time step are referred to as its *time-linked-events*. Every time that an L-costly-event is payed by other L-costly-events, it is also payed by one of its time-linked-events.

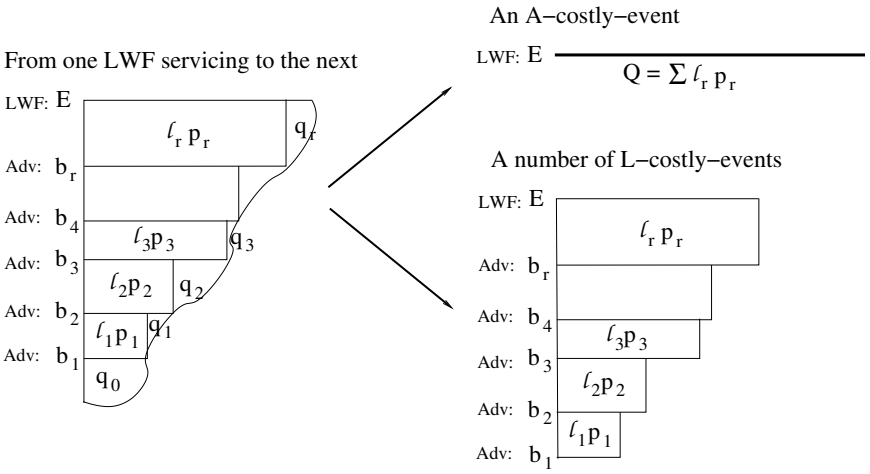

Figure 2: The left side gives the life of a page between LWF servicings. The right side gives the two ways in which the instance can be changed.

The life of a page from one servicing by LWF to the next can be depicted as a roughly triangular region as in the left side of Figure 2. The vertical axis is time and the horizontal axis is the indexes of requests. The width of the triangle increases with time as more requests arrive. Any point in the triangle is one request unserviced by LWF at one time step. The bottom point of the triangle represents the arrival of the first request after the last servicing by LWF. The top horizontal line is the time E at which LWF next services the page. The

intermediate horizontal lines are the times b_j that the adversary explicitly services the page. Wait time is measured as area. Let p_j denote the total number requests serviced at time b_j or earlier by the adversary and remaining unserviced by LWF. Let $\ell_j = b_{j+1} - e_j$ denote the amount of time until the next servicing of the page. It follows that $\ell_j p_j$ is the flow time that accumulates for LWF during this period for the requests already serviced by the adversary. Even if the adversary never services the page again, more requests can arrive for this page during this time period, because it is assumed that they will get serviced for her when LWF services the page. Let q_j denote the total flow time accumulated by LWF and the adversary during $(b_j, e_j]$ for requests for this page that arrive after the time b_j. The total wait time of this page during this time period under LWF is $\sum_{j\in[0,r]}[\ell_j p_j + q_j]$. For the adversary, the corresponding wait time for this page during this time period is $\sum_{j\in[0,r]} q_j$.

We will now transform the original instance into a canonical instance, one LWF servicing at a time. Further we will alter the rules of the game being played between LWF and the adversary. Our goal is to make a complete distinction between A-costly-events and L-costly-events. See Figure 2. Let j be an LWF servicing of some page at time E. If the adversary's flow time $\sum_{j\in[0,r]} q_j$ is at least γ of LWF's, that is at least $\gamma \sum_{j\in[0,r]}[\ell_j p_j + q_j]$, then the instance is changed so that instead $Q = \sum_{j\in[0,r]}[\ell_j p_j + q_j]$ requests arrive at time E. This is then called an *A-costly-event.*

On the other hand, if the adversary's flow time $\sum_{j\in[0,r]} q_j$ is less than $\gamma \sum_{j\in[0,r]}[\ell_j p_j + q_j]$, then we will change the instance so that any requests that arrive before the adversary services the page, are delayed until the moment the adversary next services this page. Any requests that arrive after the last adversary servicing b_r and before LWF services the page at time E, are removed from the instance. The *L-costly-events* are then the r rectangles, or time periods, $[b_j, b_{j+1}]$, $1 \le j \le r-1$, and $[b_r, E]$. The total wait time during this time period of this page under LWF decreases from $\sum_{j\in[0,r]}[\ell_j p_j + q_j]$ to $\sum_{j\in[0,r]}[\ell_j p_j]$. Under the adversary, on the other hand, it decreases from $\sum_{j\in[0,r]} q_j$ to only one time step per request. In fact, we will assume that the adversary's wait time on these requests is zero.

The above changes restrict the adversary in how she is able to choose the input instance. To compensate her, we give her the following new power. The adversary can force LWF to service any page with wait time at least $1-\gamma$ of the maximum wait time of any page. We call this the *new game.* We now show that an $O(1)$-competitiveness analysis for the new game is sufficient to establish $O(1)$-competitiveness for the original problem.

LEMMA 2.1. *If LWF is s-speed c-competitive in the new game then LWF is s-speed* $\frac{c}{\gamma}$*-competitive in the original game.*

Proof. We prove the contrapositive. Consider an adversarial strategy $\mathcal{A}$ for the original game that forces the competitiveness to be more that $\frac{c}{\gamma}$. We now explain how the adversary can mimic $\mathcal{A}$ in the new game. In the new game the adversary forces LWF to service the pages in exactly the order that $\mathcal{A}$ forced LWF to service the pages in the old game. The first thing to check is that this is a legal strategy for the adversary in the new game. This follows because for all times and all pages, in the transformation to the canonical input,

the wait time for a page at this time never increases and at most decreases by a factor of $(1-\gamma)$.

Now consider the transformation to the canonical input. In the creation of an A-costly-event, LWF's wait time does not change and the adversary's wait time increases by at most a factor of $\frac{1}{\gamma}$. Hence, the competitive ratio for servicing these requests does not decrease by more than $\frac{1}{\gamma}$. In the creation of an L-costly-event, both LWF and the adversary's wait time decrease by the same amount. Because LWF's total wait time is greater or equal to that of the adversary, it follows that this change only increases the competitive ratio. Hence, this yields an adversarial strategy for the new game that forces LWF to be at least c-competitive. ■

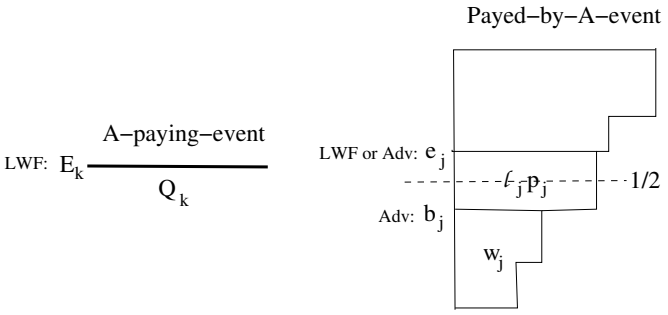

Figure 3: An A-paying-event paying a payed-by-A-event

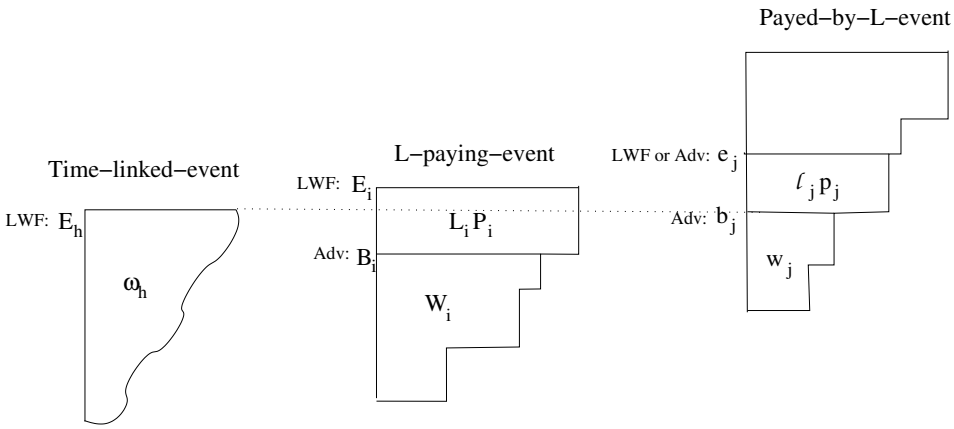

Figure 4: A time-linked-events and an L-paying-event paying a payed-by-L-event

For the rest of this section we assume a canonical input, and analyze LWF under the rules of the new game. We will now define the different types of events. Each A-costly-event needs to pay and in this role is referred to as an *A-paying-event*. See Figure 3. We will index these events by $k \in \mathcal{A}$. We denote that time that this event occurs by E_k, which is the time that all its requests arrive and are serviced by LWF. We denote by Q_k the number

of these requests, which is also equal to the wait time that these requests contribute to both to LWF and to the adversary. It follows that $\sum_{k\in\mathcal{A}} Q_k$ is the total wait time that LWF accumulates for all A-costly-events and is at most the adversary's total wait time, namely $\mathrm{OPT}_1 \geq \sum_{k\in\mathcal{A}} Q_k$.

Each L-costly-event must be payed and in this role is referred to as a *payed-event*. See Figures 3 and 4. There is one payed-event j for each time the adversary explicitly services a page. That is, there is one event for each rectangle in the bottom right of Figure 2. We will index these events by $j \in \mathcal{L}$. We denote beginning and ending times of event j by b_j and e_j, which are the time of the adversary servicing in question, and the time at which either the adversary or LWF next services the same page, respectively. Let w_j denote the flow time accumulated by LWF, for the page in question, since the last servicing of this page by LWF up to and including time b_j. As done above, let $\ell_j = b_j - e_j$ denote the duration of this period. Let p_j denote the total number requests for the page in question, that were serviced at time b_j, or earlier, by the adversary and remaining unserviced by LWF. Because the input is canonical, and j is an L-costly event, no new requests for this page arrive during (b_j, e_j). Hence, $\ell_j p_j$ is the wait time for this page that accumulates for LWF during this period. It follows that $\sum_{j\in\mathcal{L}} \ell_j p_j$ is the total wait time that LWF accumulates for all L-costly-events. Because $\sum_{k\in\mathcal{A}} Q_k$ is the total wait time accumulated by LWF for all A-costly-events, it follows that $\mathrm{LWF}_s = \sum_{j\in\mathcal{L}} \ell_j p_j + \sum_{k\in\mathcal{A}} Q_k$. Combining this with $\mathrm{OPT}_1 \geq \sum_{k\in\mathcal{A}} Q_k$, gives that $\sum_{j\in\mathcal{L}} \ell_j p_j \leq \mathrm{LWF}_s - \mathrm{OPT}_1$.

Let $j \in \mathcal{L}$ be a payed-event. During the second half of event j's life, namely $[\frac{b_j+e_j}{2}, e_j]$, LWF services a total of $\frac{s\ell_j}{2}$ pages. If at least $\epsilon s \ell_j$ of these are servicing A-paying-events, then we say that j is a *payed-by-A-event*. Otherwise, j is a payed-by-L-event. Let $\mathcal{L}_A$ be the collection of payed-by-A-events, and $\mathcal{L}_L$ the collection of payed-by-L-events.

An L-costly-event of the form $[b_r, E]$, that is, one that ends with an LWF servicing, is called an *L-paying-event*. We will index L-paying events by $i \in \widehat{\mathcal{L}}$. We denote the beginning and ending of an L-paying-event i by $B_i = b_r$ and E_i. Similarly, $L_i = B_i - E_i$, $W_i = w_r$, and $P_i = p_r$. In total, the wait time of these requests under LWF is $W_i + L_i P_i$. Because only some of the L-costly-events are included, $\sum_{i\in\widehat{\mathcal{L}}} L_i P_i \leq \mathrm{LWF}_s$.

In addition to many L-paying-events, each payed-by-L-event is payed by what we call time-linked-events. See Figure 4. *Time-linked-events* can either be A or L-costly. They focus on the total flow time LWF accumulates from one servicing of a page by LWF and the next. There is one for each time LWF services a page and we will index them by $h \in \mathcal{H}$. The ending time of time-linked-event h, denoted by E_h, is the time at which LWF services the page. Let ω_h denote the total flow time of this page under LWF since its last servicing of this page. It follows that $\sum_{h\in\mathcal{H}} \omega_h = \mathrm{LWF}_s$. There are s Time-linked-events h ending and one payed-by-L-event i beginning during any one time step $E_h = b_j$. These time-linked-events will pay this payed-by-L-event.

We now show how the main theorem will follow from the following two payment lemmas.

LEMMA 2.2. $\frac{2(s+1)}{\epsilon s} \sum_{k\in\mathcal{A}} Q_k \geq (1-\gamma) \sum_{j\in\mathcal{L}_A} \ell_j p_j$

LEMMA 2.3. $\sum_{h\in\mathcal{H}}\omega_h + \sum_{i\in\widehat{\mathcal{L}}} L_i P_i \geq (1-\gamma)^2 \cdot \frac{((1-\epsilon)s-1)^2}{2s} \sum_{j\in\mathcal{L}_L} \ell_j p_j$

Proof. (Theorem 2.1) We multiply the equation in Lemma 2.2 by $(1-\gamma)\cdot\frac{((1-\epsilon)s-1)^2}{2s}$ and add it to the equation in Lemma 2.3. Using the fact that $\mathcal{L}_A \cup \mathcal{L}_L = \mathcal{L}$, and rounding up on the left, we get that $\sum_{h\in\mathcal{H}}\omega_h + \sum_{i\in\widehat{\mathcal{L}}} L_i P_i + \frac{s}{\epsilon}\sum_{k\in\mathcal{A}} Q_k \geq (1-\gamma)^2\cdot\frac{((1-\epsilon)s-1)^2}{2s}\sum_{j\in\mathcal{L}}\ell_j p_j$ Substituting the inequalities $\sum_{h\in\mathcal{H}}\omega_h = \mathrm{LWF}_s$, $\sum_{i\in\widehat{\mathcal{L}}} L_i P_i \leq \mathrm{LWF}_s$, $\sum_{k\in\mathcal{A}} Q_i \leq \mathrm{OPT}_1$, and $\sum_{j\in\mathcal{L}_A}\ell_j p_j = \mathrm{LWF}_s - \mathrm{OPT}_1$ and setting $s=6$, $\gamma=\epsilon=0.006$ yields $\mathrm{LWF}_s + \mathrm{LWF}_s + 1{,}000\mathrm{OPT}_1 \geq 2.029(\mathrm{LWF}_s - \mathrm{OPT}_1)$, or equivalently, $\frac{\mathrm{LWF}_s}{\mathrm{OPT}_1} \leq 35{,}000$. However, we also lost at factor of $\frac{1}{\gamma}$ when we changed the instance and the rules for LWF. This gives the glorious competitive ratio of $6{,}000{,}000 = \Theta(1)$. ■

The first step toward proving Lemma 2.2 uses the order that LWF services the pages to obtain an inequality on some of the variables involved and in doing so establishes a payment from an A-paying-event to a payed-by-A-event.

LEMMA 2.4. *Let $k\in\mathcal{A}$ be an A-paying-event and $j\in\mathcal{L}_A$ be an payed-by-A-event for which LWF services in the order $b_j \lesssim E_k \lesssim e_j$. Their wait times must be $Q_k \geq (1-\gamma)(E_k - b_j)p_j$.*

Proof. LWF services the page associated with the A-paying-event k at time E_k, but because $E_k \lesssim e_j$, LWF does not service page associated with payed-by-L-event j until later. Hence, at time E_k, the page for k has wait time at least $(1-\gamma)$ of that of the page for j under LWF. At this time, the wait time on k is Q_k. The wait time on j at time E_k is $w_j + (E_k - b_j)p_j$, because at time $b_j \leq E_i$, its wait time was w_j and the p_j unserviced requests at time b_j remain unserviced during the time period $[b_j, E_k]$. It follows that $Q_k \geq (1-\gamma)[w_j + (E_k - b_j)p_j] \geq (1-\gamma)(E_k - b_j)p_j$. ■

The following lemma will establish which A-paying-events will pay which payed-by-A-event.

LEMMA 2.5. *There is setting of the 0-1 valued indicator variable $a_{k,j}$ so that if $a_{k,j}=1$ then the A-paying-event k occurs during the second half of the payed-by-A-event j's life, namely $E_k \in [\frac{b_j+e_j}{2}, e_j]$, each k does not pay to more than $\sum_{j\in\mathcal{L}_A} a_{k,j} \leq \frac{s+1}{\epsilon s}$ different j's, and each j is payed by at least $\sum_{k\in\mathcal{A}} a_{k,j} \geq 1$.*

We are now ready to compute the total payment from A-paying-events k to payed-by-A-event j.

Proof. (Lemma 2.2) Using the indicator variables $a_{k,j}$ from Lemma 2.5 as coefficients in a linear combination of the inequalities from Lemma 2.4, gives the needed result. That is, $\sum_{k\in\mathcal{A}, j\in\mathcal{L}_A} a_{k,j}[Q_k] \geq \sum_{k\in\mathcal{A}, j\in\mathcal{L}_A} a_{k,j}[(1-\gamma)(E_k - b_j)p_j]$, and $\sum_{k\in\mathcal{A}}\frac{s+1}{\epsilon s}[Q_k] \geq \sum_{j\in\mathcal{L}_A}[(1-\gamma)(E_k - b_j)p_j]$. The last step is to note that when $a_{k,j}=1$ we have $E_k \in [\frac{b_j+e_j}{2}, e_j]$ and hence $E_k - b_j \geq \frac{\ell_j}{2}$. ■

Proof. (Lemma 2.5) We form a bipartite graph with $\frac{s+1}{\epsilon s}$ copies of each A-paying-event k on one side and each payed-by-A-event j on the other side. There is an edge $\{k, j\}$ if $E_k \in [\frac{b_j+e_j}{2}, e_j]$. We use Hall's theorem is establish that this graph has a matching that matches each paid-by-A-event. Then we will set $a_{k,j}$ equal to 1 if and only if edge $\{k, j\}$ is in this matching.

Hall's theorem states that a sufficient condition for there to exist a matching that covers the paid-by-A-events is that for each subset S of the paid-by-A-events it is the case that $|S| \leq |N(S)|$, where $N(S)$ is the collection of vertices adjacent to a vertex in S. Consider such an S. We consider the intervals $[\frac{b_j+e_j}{2}, e_j]$, for $j \in S$. Let ℓ_S be the number of time steps that these intervals cover, namely $\ell_S = |\cup_{j \in S}[\frac{b_j+e_j}{2}, e_j]|$. We have that $|S| \leq (s+1)\ell_S$ because at the end e_j of a payed-by-A-event, either LWF or the adversary services some request.

We now explain how to find a mutually disjoint subcollection of these intervals with measure at least $\ell_S/2$. Break ℓ_S into connected components, i.e. that cover time steps without gap. For each component, start by taking the interval that starts earliest in time. Then repeat the following until this connected component is covered: Amongst the intervals that overlap with previous intervals select the one that ends latest. The resulting intervals cover ℓ_S. It is easy to verify that the intervals selected in the odd (even) iterations are mutually disjoint. Either the intervals taken in the odd iterations cover $\ell_S/2$ units of time, or the intervals taken in the even iterations cover $\ell_S/2$ units of time.

Now consider a selected interval of length ℓ_j. Since this interval is the top half of a payed-by-A-event, it contains at least $\epsilon s \ell_j$ A-paying events. Hence our selected intervals cover at least $\epsilon s \ell_S$ A-paying-events. All of these A-paying-events are contained in the neighborhood $N(S)$ of S copied $\frac{s+1}{\epsilon s}$ times each, giving that $N(S)$ is at least $\frac{s+1}{\epsilon s} \cdot (\epsilon s \ell_S) = (s+1)\ell_S$. Combining $|S| \leq (s+1)\ell_S$, and $(s+1)\ell_S \leq |N(S)|$, gives that $|S| \leq |N(S)|$. ■

This completes our analysis of payments from A-paying-events to payed-by-A-events. We now will analyze the payments from time-linked-events and L-paying-events to a payed-by-L-events. As done before, the first step toward proving Lemma 2.3 is to establish a payment scheme.

LEMMA 2.6. *Let $h \in \mathcal{H}$ be an time-linked-event, $i \in \widehat{\mathcal{L}}$ be an L-paying-event, and $j \in \mathcal{L}_L$ be an payed-by-L-event for which LWF services in the order $B_i \leq b_j = E_h \lesssim E_i \lesssim e_j$. Their wait times must satisfy $\omega_h + (E_i - b_j)P_i \geq (1-\gamma)^2(E_i - b_j)p_j$.*

Proof. LWF services the time-linked-event h at time $E_h \lesssim E_i$, before it services the L-paying-event. At time E_h, the wait time on h is ω_h and on i is $W_i + (E_h - B_i)P_i$. Hence according to the new rules for LWF. It follows that $\omega_h \geq (1-\gamma)[W_i + (E_h - B_i)P_i]$ or that $\frac{\omega_h}{1-\gamma} - (E_h - B_i)P_i \geq W_i$. Similarly, LWF services the L-paying-event i at time $E_i \lesssim e_j$, before it services the payed-by-L-event j. At time E_i, the wait time on i is $W_i + L_i P_i$ and on j is $w_j + (E_i - b_j)p_j$. It follows that $W_i + L_i P_i \geq (1-\gamma)[w_j + (E_i - b_j)p_j]$. Substituting,

we get $\frac{\omega_h}{1-\gamma} - (E_h - B_i)P_i + L_i P_i \geq (1-\gamma)[w_j + (E_i - b_j)p_j]$. By dropping the w_j term and noticing that that $L_i - (E_h - B_i) = (E_i - B_i) - (E_h - B_i) = (E_i - E_h) = (E_i - b_j)$, we obtain the desired result $\omega_h + (E_i - b_j)P_i \geq (1-\gamma)^2(E_i - b_j)p_j$. ■

Now Lemma 2.3 follows by taking a linear combination of the inequalities from Lemma 2.6.

Proof. (Lemma 2.3) For each L-paying-event $i \in \widehat{\mathcal{L}}$ and each payed-by-L-event $j \in \mathcal{L}_L$, define $a_{i,j}$ as follows:

$$i \in \widehat{\mathcal{L}}, j \in \mathcal{L}_L, a_{i,j} = \begin{cases} \frac{1}{\ell_j} & \text{if } B_i \leq b_j \lesssim E_i \lesssim e_j \\ 0 & \text{otherwise} \end{cases}$$

Using this both as a scaling factor for the equations from Lemma 2.6 and as an indicator variable as to when the equation holds, gives the following for all $i \in \widehat{\mathcal{L}}$, $j \in \mathcal{L}_L$, and $h \in \mathcal{H}$ for which $b_j = E_h$: $a_{i,j}[\omega_h + (E_i - b_j)P_i] \geq a_{i,j}[(1-\gamma)^2(E_i - b_j)p_j]$. Since LWF services s pages at time b_j, there are s time-linked-events for which $b_j = E_h$. Of these, let h_j denote the one with the smallest ω_h. Summing over all i and j, the above equation gives $\sum_{i \in \widehat{\mathcal{L}},\ j \in \mathcal{L}_L} a_{i,j}\omega_{h_j} + \sum_{i \in \widehat{\mathcal{L}},\ j \in \mathcal{L}_L} a_{i,j}(E_i - b_j)P_i \geq (1-\gamma)^2 \sum_{i \in \widehat{\mathcal{L}},\ j \in \mathcal{L}_L} a_{i,j}(E_i - b_j)p_j$. The lemmas 2.7, 2.8, and 2.9 below, then compare each of these sums with the required terms in Lemma 2.3 giving as required that $\sum_{h \in \mathcal{H}} \omega_h + \sum_{i \in \widehat{\mathcal{L}}} L_i P_i + \geq (1-\gamma)^2 \cdot \frac{((1-\epsilon)s-1)^2}{2s} \sum_{j \in \mathcal{L}_L} \ell_j p_j$. ■

LEMMA 2.7. $\sum_{i \in \widehat{\mathcal{L}},\ j \in \mathcal{L}_L} a_{i,j}\omega_{h_j} \leq \sum_{h \in \mathcal{H}} \omega_h$

Proof. Let $j \in \mathcal{L}_L$ be any payed-by-L-event. There are at most $s(e_j - b_j) = s\ell_j$ L-paying-events $i \in \widehat{\mathcal{L}}$ for which $b_j = E_h \lesssim E_i \lesssim e_j$, because at each of the time steps $b_j \lesssim t \lesssim e_j$, LWF services at most s pages. The number of L-paying-events i for which $a_{i,j}$ is non-zero is thus at most $s\ell_j$. When it is non-zero, $a_{i,j} = \frac{1}{\ell_j}$. This gives that $\sum_{i \in \widehat{\mathcal{L}}} a_{i,j} \leq s$.

At time b_j, when the page associated with payed-by-L-event j is being serviced by the adversary, there are s pages serviced by LWF. Let the wait times of the corresponding time-linked-events be $\omega_{h_{j,1}}, \ldots, \omega_{h_{j,s}}$. Each of these is at least ω_{h_j}, this being by definition of h_j. If follows that $\sum_{h \in \mathcal{H}} \omega_h \geq \sum_{j \in \mathcal{L}_L} (\omega_{h_{j,1}} + \ldots + \omega_{h_{j,s}}) \geq \sum_{j \in \mathcal{L}_L} s\omega_{h_j} \geq \sum_{j \in \mathcal{L}_L} (\sum_{i \in \widehat{\mathcal{L}}} a_{i,j})\omega_{h_j}$, as required for the lemma. ■

LEMMA 2.8. $\sum_{i \in \widehat{\mathcal{L}},\ j \in \mathcal{L}_L} a_{i,j}(E_i - b_j)P_i \leq \sum_{i \in \widehat{\mathcal{L}}} L_i P_i$

Proof. Let $i \in \widehat{\mathcal{L}}$ be some L-paying-event. There are at most $L_i = E_i - B_i$ payed-by-L-events $j \in \mathcal{L}_L$ for which $B_i \leq b_j \lesssim E_i$, because at each of the time steps $B_i \leq t \lesssim E_i$, the adversary services at most one page. The number of j for which $a_{i,j}$ is non-zero is at most L_i. When $a_{i,j}$ is non-zero $E_i \lesssim e_j$. Hence, $a_{i,j}(E_i - b_j) = \frac{1}{\ell_j}(E_i - b_j) \leq \frac{1}{(e_j - b_j)}(e_j - b_j) = 1$. This gives that $\sum_{i \in \widehat{\mathcal{L}}} \sum_{j \in \mathcal{L}_L} a_{i,j}(E_i - b_j)P_i \leq \sum_{i \in \widehat{\mathcal{L}}} L_i \cdot 1 \cdot P_i$, as required for the lemma. ■

LEMMA 2.9. $\sum_{i\in\widehat{\mathcal{L}},\ j\in\mathcal{L}_L} a_{i,j}(E_i - b_j)p_j \geq \frac{((1-\epsilon)s-1)^2}{2s}\sum_{j\in\mathcal{L}_L}\ell_j p_j$

Proof. Let $j \in \mathcal{L}_L$ be any payed-by-L-event. There are potentially $s(e_j - b_j) = s\ell_j$ L-paying-events $i \in \widehat{\mathcal{L}}$ for which $b_j \lesssim E_i \lesssim e_j$, because at each of the time steps, $b_j \lesssim t \lesssim e_j$, LWF services s pages. However, during some of these servicings LWF may be servicing A-paying-events, but by the definition of this payed-event j being a payed-by-L-event, it does so at most $\epsilon s\ell_j$ times. As well, for $a_{i,j}$ to be non-zero, we need that $B_i \leq b_j \lesssim E_i \lesssim e_j$. The number for which $b_j \lesssim B_i \lesssim e_j$ is at most $e_j - b_j = \ell_j$, because the adversary services at most one page each of these points in time. Subtracting off the at most $\epsilon s\ell_j$ which might be A-paying-events and the at most ℓ_j for which $b_j < B_i < E_i \lesssim e_j$ from the potentially $s\ell_j$ for which $b_j \lesssim E_i \lesssim e_j$, gives that there are at least $((1-\epsilon)s-1)\ell_j$ for which $B_i \leq b_j \lesssim E_i \lesssim e_j$ and hence $a_{i,j}$ for which non-zero.

For each of these i, our goal is to bound the sum $\sum_{i\in\widehat{\mathcal{L}}} a_{i,j}(E_i - b_j)p_j = \frac{1}{\ell_j}p_j\sum_{i\in\widehat{\mathcal{L}},a_{i,j}\neq 0}(E_i - b_j)$. In the worst case, each $E_i - b_j$ is as small as it can be. Because $b_j \lesssim E_i$, we know that $E_i - b_j \geq 0$. However, at most s of them can have any one particular value t, because at most s pages are serviced by LWF at time $E_i = b_j + t$. Hence, in the worst case there are s L-paying-events i for which $E_i - b_j$ is 1, s for which it is 2, s for which it is 3, ... We have seen that there are at least $((1-\epsilon)s-1)\ell_j$ of them, so this sequence ends with s of them for which $E_i - b_j$ is $((1-\epsilon)s-1)\ell_j/s$. In this worst case, $\frac{1}{\ell_j}p_j\sum_{i\in\widehat{\mathcal{L}},a_{i,j}\neq 0}(E_i - b_j)$ is $\frac{1}{\ell_j}p_j s\sum_{t\in[1,((1-\epsilon)s-1)\ell_j/s]} t \geq \frac{1}{\ell_j}p_j s[((1-\epsilon)s-1)\ell_j/s]^2/2 = \frac{((1-\epsilon)s-1)^2}{2s}\ell_j p_j$. This gives the required result $\sum_{i\in\widehat{\mathcal{L}},\ j\in\mathcal{L}_L} a_{i,j}(E_i - b_j)p_j \geq \frac{((1-\epsilon)s-1)^2}{2s}\sum_{j\in\mathcal{L}_L}\ell_j p_j$. ■

3 Lower Bound

THEOREM 3.1. *There exists instances I such that*

$$\frac{\mathrm{LWF}_s(I)}{\mathrm{OPT}_1(I)} = \Omega\left(\frac{n^{1-\frac{\ln s}{\ln(1+1/s)}}}{\ln n}\right)$$

which is $n^{\Omega(1)}$ for $s < \frac{1+\sqrt{5}}{2} \approx 1.618$

Proof. We now define the instance I. We encourage the reader to refer to figure 3 to help understand the construction. The instance I is partitioned into R sets $S_0, S_1, S_2, \ldots, S_R$ of requests, where $R = \log_{1+1/s} n = \frac{\ln n}{\ln(1+1/s)}$. The set of requests S_r will contain requests for $n_r = \frac{n}{s^r}$ different pages. All requests for a particular page in S_r arrive at the same time. Thus a total of $\Theta(n)$ pages are requested.

S_0 consists of the one request arriving at time zero for each of n different pages $P_1, \ldots P_n$. We now define the remaining S_r sets of requests. Let $T_r = n_1 + n_2 + n_3 + \ldots + n_r$. Let $w_r = \frac{1}{s}(1+\frac{1}{s})^{r-1}n$; think of w_r as a common wait time. For each $r \in [1, R]$ and $i \in [0, n_r)$, request $R_{r,i} \in S_r$ will have $p_{r,i} = \frac{w_r}{i}$ requests arrive at time $T_r - i$ for page P_{T_r-i}. So note that the n_1 requests in S_1 are to the same pages as the first n_1 requests in S_0, and

the first $n - n_1$ requests in S_2 are to the same pages as the last n_1 requests in S_0. After this, no page is requested at more than one time. The fact that $w_R/R > 1$ follows since $s < 1 + 1/s$ for $s < \frac{1+\sqrt{5}}{2}$. Note that at time T_r, all of the pages in S_r have total wait time of $p_{r,i} \cdot ([T_r] - [T_r - i]) = \frac{w_r}{i} \cdot i = w_r$.

We now give a bit of intuition and explanation, and then formally prove that our explanation is correct. The ith collection of requests from the S_r's, $r \geq 1$, arrives at time i, and the adversary immediately finishes these requests. So the adversary's wait time on these requests is equal to the number of such requests. LWF first does the requests in S_0 in order, finishing these at time $n_1 = T_= n/s$. Then during each time period $[T_r, T_{r+1}]$, $r \geq 1$, while the adversary is processing S_{r+1}, LWF is processing the requests in S_r in the reverse order of their arrival.

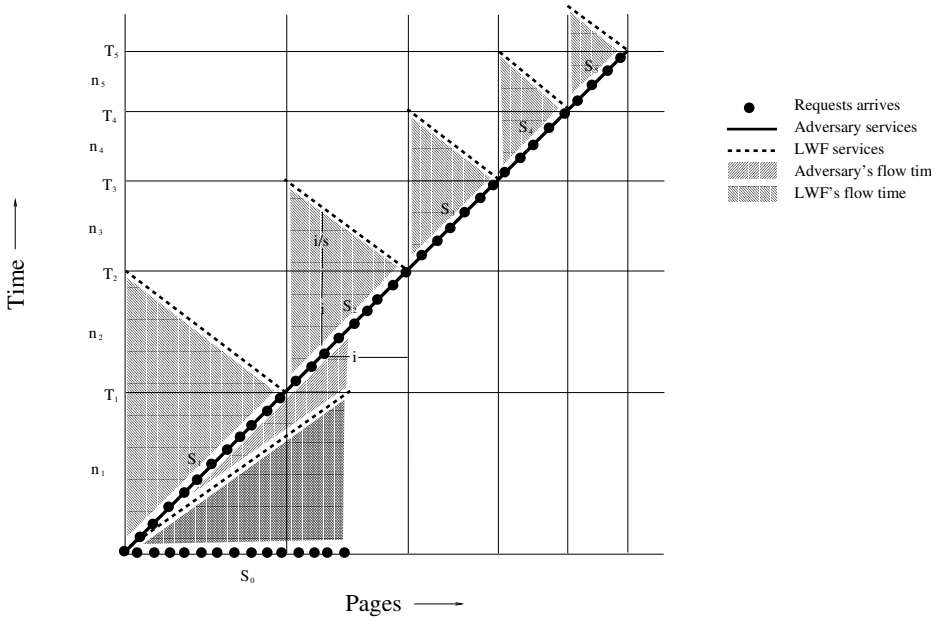

Figure 5: The time that requests for requests arrive and are serviced by LWF and by the adversary.

We will now prove that LWF services the requests in the stated order as indicated in Figure 3. Specifically, LWF services request $R_{0,i}$ at time $\frac{i}{s}$ and request $R_{r,j}$ at time $T_r + \frac{i}{s}$.

The requests in S_0 are equivalent until new requests come, so we can decide the order that they get completed. In order to prove that all requests in S_0 get serviced before those in S_1, consider requests $R_{0,i} \in S_0$ for $i \in [0, n)$ and $R_{1,j} \in S_1$ for $j \in [0, n_1)$. At time $\frac{i}{s}$, when $R_{0,i}$ is stated to be serviced, its one request has wait time $\frac{i}{s}$. At this time, $R_{1,j}$'s $\frac{w_1}{j}$ requests have age $[\frac{i}{s}] - [T_1 - j]$ and wait time $[\frac{i}{s} - T_1 + j] \cdot w_1 \frac{1}{j} = [\frac{i}{s} - \frac{n}{s} + j] \cdot \frac{n}{s} \frac{1}{j} = -\frac{n-i}{s} \cdot \frac{n}{sj} + \frac{n}{s}$. Note that $i < n$ and $j < n_1 = \frac{n}{s}$ giving that $\frac{n}{sj} > 1$. Hence, $R_{1,j}$'s wait time is less than $-\frac{n-i}{s} + \frac{n}{s} = \frac{i}{s}$, which is $R_{0,i}$'s wait time. Hence, LWF chooses to service $R_{0,i}$ over $R_{1,j}$.

In order to prove that LWF services the requests in S_r in the order $i = 0, 1, 2, \ldots, n_r - 1$, consider requests $R_{r,i}$ and $R_{r,j}$ for $0 \leq i < j < n_r$. At time $T_r + \frac{i}{s}$, when $R_{r,i}$ is stated to be serviced, its $\frac{w_r}{i}$ requests have age $[T_r + \frac{i}{s}] - [T_r - i] = [(\frac{1}{s} + 1)i]$ and wait time $[(\frac{1}{s}+1)i] \cdot w_r \frac{1}{i} = (\frac{1}{s}+1)w_r$. At this time, $R_{r,j}$'s $\frac{w_r}{j}$ requests have age $[T_r + \frac{i}{s}] - [T_r - j] = [\frac{i}{s} + j]$ and wait time $[\frac{i}{s} + j] \cdot w_r \frac{1}{j} < [\frac{j}{s} + j] \cdot w_r \frac{1}{j} = (\frac{1}{s} + 1) w_r$. Hence, LWF chooses to service $R_{r,i}$ over $R_{r,j}$.

In order to prove that all requests in S_r get serviced before those in S_{r+1}, consider requests $R_{r,i} \in S_r$ for $i \in [0, n_r)$ and $R_{r+1,j} \in S_{r+1}$ for $j \in [0, n_{r+1})$. At the time $T_r + \frac{i}{s}$ when $R_{r,i}$ is serviced, $R_{r+1,j}$'s $\frac{w_{r+1}}{j} = \frac{w_r}{j}(1 + \frac{1}{s})$ requests have age $[T_r + \frac{i}{s}] - [T_{r+1} - j] = [\frac{i}{s} - n_{r+1} + j] = [\frac{i - n_r}{s} + j]$ and wait time $[\frac{i-n_r}{s} + j] \cdot \frac{w_r}{j}(1 + \frac{1}{s}) < [j] \cdot \frac{w_r}{j}(1 + \frac{1}{s}) = w_r(1 + \frac{1}{s})$. This being the wait time of $R_{r,i}$, LWF chooses to service $R_{r,i}$ over $R_{r,j}$. This completes the proof that LWF services the requests at the stated times.

We are now ready to compute LWF's flow time. As we have seen request $R_{0,i}$ has wait time $\frac{i}{s}$ upon completion for a total flow from S_0 of $\frac{n^2}{2s}$ and request $R_{r,i}$ has wait time $w_r(1 + \frac{1}{s})$ for a total flow from S_r of $n_r \cdot w_r(1 + \frac{1}{s}) = \frac{n}{s^r} \cdot \frac{1}{s}(1 + \frac{1}{s})^r n = \frac{1}{s}(\frac{1}{s} + \frac{1}{s^2})^r n^2$. LWF's speed s is restricted to be less than $\frac{1+\sqrt{5}}{2}$ so that $(\frac{1}{s} + \frac{1}{s^2}) > 1$ and hence the wait time for S_r grows exponentially with r. At any rate, we will bound LWF's flow time by only this last term, giving that $\mathrm{LWF}_s(I) \geq \Omega(n_R w_R)$.

Similarly, we compute the adversary's flow time $\mathrm{OPT}_1(I)$ as follows. The one request for $R_{0,i}$ has wait time i upon completion for a total flow from S_0 of $\frac{n^2}{2}$. The adversary services the remaining requests as they arrive, so incurs a flow of one for each. $R_{r,i}$ has $\frac{w_r}{i}$ requests for a total flow from S_r of $\sum_{i \in [0, n_r)} \frac{w_r}{i} = w_r \ln n_r$. Because $w_r = \frac{1}{s}(1 + \frac{1}{s})^{r-1} n$, this flow time grows exponentially with r, giving that the sum of these terms for $r \in [1, R]$ is dominated by the last term, giving that $\mathrm{OPT}_1(I) = \frac{n^2}{2} + O(w_R \ln n_R)$. The number new requests continues to increase with time until their flow dominates that of the common $\Theta(n^2)$ flow time. This is achieved by setting $R = \frac{\ln n}{\ln(1+1/s)}$. This gives $w_R = \Omega\left((1 + \frac{1}{s})^R n\right) = \Omega\left((1 + \frac{1}{s})^{\frac{\ln n}{\ln(1+1/s)}} n\right) = \Omega(n^2)$. In conclusion, we also bound the adversary's flow time by only its last term, giving that $\mathrm{OPT}_1(I) \leq O(w_R \ln n_R)$.

The competitive ratio is then easily computed to be $\frac{\mathrm{LWF}_s(I)}{\mathrm{OPT}_1(I)} = \Omega\left(\frac{n_R w_R}{w_R \ln n_R}\right) = \Omega\left(\frac{1}{\ln n} \frac{n}{s^R}\right) = \Omega\left(\frac{1}{\ln n} \frac{n}{s^{\frac{\ln n}{\ln(1+1/s)}}}\right) = \Omega\left(\frac{1}{\ln n} n^{1 - \frac{\ln s}{\ln(1+1/s)}}\right)$. ■

4 Conclusion

The obvious open question is to close the gap between the upper and lower bounds of the speed required for LWF to be $O(1)$-competitive. The upper bound analysis and the lower bound construction match in many ways, for example, the worst case for LWF is if A-costly-events happen early followed by only L-costly events. Our guess is that the lower bound is either tight, or at least closer to being tight than the upper bound. By allowing non-integer speeds, the upper bound on the speed that this analysis provides is $(5.83 + \epsilon)$. We seem to

have an upper bound analysis with speed $s = 4.69 + \epsilon$, but the proof complexity goes up substantially.

References

[1] S. Aacharya, and S. Muthukrishnan, "Scheduling on-demand broadcasts: new metrics and algorithms", ACM/IEEE Int. Conf. on Mobile Computing and Networking, 43 – 54, 1998.

[2] D. Aksoy, and M. Franklin, "Scheduling for large-scale on-demand data broadcasting", IEEE INFOCOM, 651-659, 1998.

[3] M. H. Ammar, and J. W. Wong, "The design of teletext broadcast cycles", *Performance Evaluation*, **5**(4), 235–242, 1985.

[4] Nikhil Bansal, personal communication.

[5] Y. Bartal, and S. Muthukrishnan, "Minimizing maximum response time in scheduling broadcasts", ACM/SIAM Symposium on Discrete Algorithms, 558 – 559, 2000.

[6] P. Chrysanthis, V. Liberatore, and K. Pruhs, "Middleware support for multicast-based data dissemination: A working reality", IEEE Workshop on Reliable Dependable Systems, 2003.

[7] J. Beaver, W. Li, V. Penkrot, S. Roychowdhury, M. Sharaf, W. Zhang, P. Chrysanthis, K. Pruhs and V. Liberatore, "An optimized multicast based data dissemination middleware: a demonstration", IEEE International Conference on Data Engineering, 2003.

[8] DirecPC website, `http://www.direcpc.com`.

[9] H.D. Dykeman, M. Ammar, and J.W. Wong, "Scheduling algorithms for videotex systems under broadcast delivery" IEEE International Conference on Communications, 1986.

[10] J. Edmonds, "Scheduling in the dark", *Theoretical Computer Science*, **235**(1), 109 – 141, 2000.

[11] J. Edmonds and K. Pruhs, "Multicast pull scheduling: when fairness is fine", *Journal of Algorithms* **36**(3), 315 – 330, 2003.

[12] T. Erlebach and A. Hall, "Hardness of broadcast scheduling and inapproximability of single-source unsplittable min-cost flow", *Journal of Scheduling*, **7**, 223 – 241, 2004.

[13] R. Gandhi, S. Khuller, Y. Kim and Y-C. Wan, "Approximation algorithms for broadcast scheduling", *Algorithmica*, **38**, 597- 608, 2004.

[14] R. Gandhi, S. Khuller, S. Parthasarathy, A. Srinivasan, "Dependent rounding in bipartite graphs," IEEE Conference on Foundations of Computer Science, 2002.

[15] A. Hall and H. Täubig, "Comparing push and pull-based broadcasting, or: would "Microsoft watches" profit from a transmitter?" Workshop on Experimental and Efficient Algorithms, 2003.

[16] B. Kalyanasundaram, and K. Pruhs, "Speed is as powerful as clairvoyance", *Journal of the ACM*, **47**(4), 617 – 643, 2000.

[17] B. Kalyanasundaram, K. Pruhs, and M. Velauthapillai, "Scheduling broadcasts in wireless networks", *Journal of Scheduling*, **4**(6), 339 – 354, 2000.

[18] H. Kaplan, R. Tarjan, K. Tsioutsiouliklis, "Faster kinetic heaps and their use in broadcast scheduling" ACM/SIAM Symposium on Discrete Algorithms, 2001.

[19] C. Kenyon, N. Schabanel and N. Young, "Polynomial-time approximation schemes for data broadcast", ACM Symposium on Theory of Computing, 659-666, 2000.

[20] C. Phillips, C. Stein, E. Torng, and J. Wein "Optimal time-critical scheduling via resource augmentation", *Algorithmica*, **32**(2), 163 – 200, 2002.

[21] K. Pruhs, J. Sgall, and E. Torng, "Online Scheduling", in *Handbook of Scheduling: Algorithms, Models and Performance Analysis*, CRC press, editors J. Leung and J. Anderson, 2004.

[22] K. Pruhs and P. Uthaisombut, "A comparison of multicast pull models", European Symposium on Algorithms, 2002.

[23] M.W. Wong, "Broadcast delivery", Proceedings of the IEEE, 1566–1577, 1988.
[24] J. Xu, D. L. Lee, Q. Hu, and W.-C. Lee. "Data Broadcast", Handbook of Wireless Networks and Mobile Computing, Ivan Stojmenovic, Ed., John Wiley, 243–265, 2002.