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

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 $\max_I \frac{A_s(I)}{\mathrm{OPT}_1(I)} \leq$

[*]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.

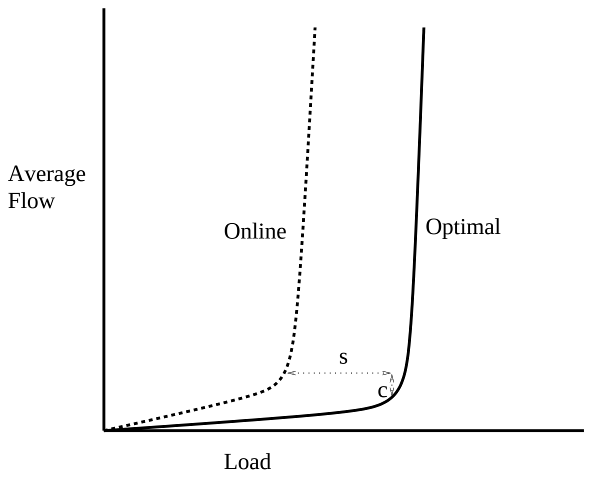

Figure 1: The worst possible performance curve of an s-speed c-competitive online algorithm.

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. An online algorithm A is locally c-competitive if at all times the increase in the objective function for A increases by at most c times the rate that the objective function increases for any adversary [21]. Building on the work in [10], in [11] it was shown that there exists an algorithm that is $(4+\epsilon)$-speed $O(1+1/\epsilon)$-competitive. Unfortunately, this algorithm is not particularly natural, and it is known that it 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.

Traditionally the multicast pull problem for constant sized pages has been formalized in a discrete time model. However, when one considers variable speed servers, there is some intuitive appeal to formalizing the problem in a continuous time model. In particular, the discrete time model is not so natural for non-integer speeds. In this paper we prove our upper bound in a discrete time model and our lower bound in a continuous time model. Our lower bound on speed being non-integer makes it natural for us to adopt the continuous model there. We adopt the discrete model for our integer upper bound on speed because we believed that this would simplify the algebraic complexity of the argument. We believe that our analysis are valid for both the discrete and continuous time models. But, strictly speaking, the results in this paper do not formally establish this.

1.2 Related Results

In order to obtain a positive result for multicast pull scheduling for unit sized pages, [17] resorted to considering offline algorithms. They found an LP-based polynomial-time algorithm that is 3-speed 3-approximate. Subsequently better results were obtained by using different rounding schemes. In [12] a rounding that is 6-speed 1-approximate algorithm is given. In [13] a rounding that is 2-speed 2-approximate is given. In [14] a rounding that is 3-speed 1-approximate is given. Finally, in [4], a rounding that is $O(1+\epsilon)$-speed $O(1)$-approximate 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, 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]. For a survey on online scheduling, including multicast pull scheduling, see [21]. Problems that are special cases of multicast pull scheduling include weighted flow time, and scheduling jobs with sequential/parallel speed-up curves [11, 21].

LWF can be implemented in logarithmic time per query [18].

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. A total of Q requests for these pages arrive over time.

Discrete Time Model: Events at each integer time t happen in the following order:

1. First, the s-speed server for LWF decides on up to s pages to broadcast.
2. Then the unit speed adversary decides on a page to broadcast.
3. And finally the server receives $R_i(t) \geq 0$ new requests for page $i \in \mathcal{P}$ (for each $1 \leq i \leq n$).

It is important to remember this order as it will be implicitly used throughout our analysis. We say that the $R_i(t)$ requests for page i at time t are *satisfied/serviced/completed* at time $C_i(t)$ if $C_i(t) > t$ is the first time when page i is broadcast after these requests arrived. 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 $C_i(t) - t$, which is how long a client that requested page i at time t has to wait until page i is broadcast. The *total response time* is then $\sum_t \sum_i R_i(t) \cdot (C_i(t) - t)$.

LWF Definition: 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 $\sum_{\hat{C}_i(t) \leq t' < t} R_i(t') \cdot (t - t')$. At each time t, LWF broadcasts the up to s pages with largest wait. It will be convenient in our 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. Ties may be broken arbitrarily.

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 a broadcast by LWF or the adversary. We then use the notation $S \lesssim T$ to mean that S happened before T in the scheduler's time. In contrast, $S < T$ means that the S happened in an strictly earlier time step. If for example, S is the broadcast of a page i at time t by LWF and T is the broadcast of a page j at time t by the adversary then $S \lesssim T$, but $S \not< T$ and $T \,/\!\!\lesssim S$. As another example, S is the broadcast of a page i at time t by LWF and T is the broadcast of a page j at time t by LWF then $S \lesssim T$ is equivalent to saying the i's wait is at least j's wait at time t.

Continuous Time Model: In this model, requests for pages may arrive at arbitrary real times. A speed s processor picks a page to broadcast at each time that is an integer multiple of $\frac{1}{s}$. A page that a speed s processors starts to broadcast at time t, completes at time $t + \frac{1}{s}$. All other definitions, and the description of LWF are as in the discrete time model.

2 Analysis of LWF

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

Theorem 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.

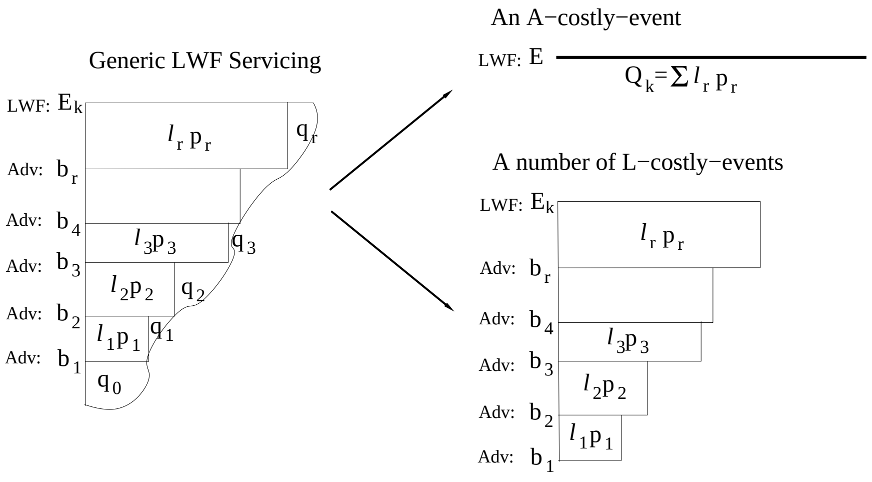

Figure 2: The figure on the left shows a generic lifetime for requests satisfied by a particular LWF broadcast. The two figures on the right side show the two possible results of the canonical transformations.

Consider a collection Q_k of requests for a particular page satisfied by an LWF servicing at time E_k. The lifetime of these requests 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 indices of requests. The width of the triangle increases with time as more requests arrive. Any point in the triangle is represents a point in time for a request between its release and its servicing by LWF. The bottom point of the triangle represents the arrival of the first request in Q_k. The top horizontal line is the time E_k at which LWF next services the requests in Q_k. The intermediate horizontal lines are the times b_j, $1 \leq j \leq r$, that the adversary explicitly services the page. Let $e_j = b_{j+1}$ for $1 \leq j \leq r-1$, and

$e_r = E$. Wait time is measured as area. Let p_j, $1 \leq j \leq r$, denote the total number of requests serviced at time b_j or earlier by the adversary and remaining unserviced by LWF. Let $\ell_j = e_j - b_j$, $1 \leq j \leq r$, 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 $[b_r, E_k]$, because of our assumption that they will get serviced for her when LWF services the page at time E_k. Let q_j, $1 \leq 1 \leq r$ denote the total flow time accumulated by LWF and the adversary during $[b_j, e_j]$ for requests for this page that arrive during $[b_j, e_j - 1]$. Let q_0 denote the total flow time accumulated by the adversary for the pages that she broadcasts at time b_1. The total wait time of this page during this time period under LWF is $q_0 + \sum_{j\in[1,r]} (\ell_j p_j + q_j)$. 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. We now describe how to create a canonical input.

Canonical Input: Let γ be some small constant that we will specify later. A canonical input is constructed by repeating the following construction for each LWF servicing k of a collection Q_k of requests for some page at time E_k. We consider two cases depending up how the adversary's cost minus Q_k, $\sum_{j\in[0,r]} q_j - Q_k$, compares to LWF's cost, $\left(q_0 + \sum_{j\in[1,r]} (\ell_j p_j + q_j)\right)$.

Case of an A-costly-event: If $\sum_{j\in[0,r]} q_j - Q_k \geq \gamma \left(q_0 + \sum_{j\in[1,r]} (\ell_j p_j + q_j)\right)$, then the instance is changed so that the Q_k requests under consideration are replaced by $\left(q_0 + \sum_{j\in[1,r]} (\ell_j p_j + q_j)\right)$ requests for this page at time $E_k - 1$. This is then called an *A-costly-event.* Thus LWF's cost for these requests remain unchanged, and the adversary's cost increases to LWF's cost. Because the competitive ratio we are proving is 6, this same cost is 6 times costlier for the adversary. Hence, the name A-costly event.

Case of an L-costly-event: If $\sum_{j\in[0,r]} q_j - Q_k < \gamma \left(q_0 + \sum_{j\in[1,r]} (\ell_j p_j + q_j)\right)$, then we will change the instance in the following way. If a request in Q_k arrives before time b_1, its arrival is delayed until time $b_1 - 1$. If a request in Q_k arrives during the interval $[b_j, b_{j+1} - 1]$, $1 \leq j \leq r - 1$, then its arrival is delayed until time $b_{j+1} - 1$. If a request in Q_k arrives at time b_r or later, it will be removed from the instance. The *L-costly-events* are then the r rectangles, or time periods, $(b_j, b_{j+1}]$, $1 \leq j \leq r - 1$, and $(b_r, E]$. The total wait time for LWF for the requests in Q_k decreases from $\left(q_0 + \sum_{j\in[1,r]} (\ell_j p_j + q_j)\right)$ to $Q_k + \sum_{j\in[1,r]} [\ell_j p_j]$. The total wait time for the adversary for the requests in Q_k decreases from $\sum_{j\in[0,r]} q_j$ to Q_k

We now define what we mean by the new game.

New Game: The adversary must give LWF a canonical input. To compensate the adversary, she is allowed to force LWF to service any page with wait time at least $1 - \gamma$ of the maximum wait time of any page. The adversary still implicitly completes a page when LWF services the page.

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 *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 $\mathcal{X}$ for the original game that forces the competitiveness to be more that $\frac{c}{\gamma}$. We now explain how the adversary can mimic $\mathcal{X}$ in the new game. In the new game the adversary forces LWF to service the pages in exactly the order that $\mathcal{X}$ 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. We need to argue that our transformation never increases the wait time of a page, and at the time E_k that a page is broadcast in the old game, the wait time for this page in the new game is at least $1-\gamma$ of the wait time of that page in the old game. That wait times never increase is obvious since both canonical transformations only delay requests. To see that wait times are not decreased too much at broadcast time, consider a page with wait time W that LWF broadcast at time E_k in the old game. It is obvious that the canonical transformation that produces A-costly-events doesn't decrease the wait time of the page at time E_k. So assume that k is a L-costly-event. In this case, LWF's wait at time E_k decreases by $\sum_{j\in[0,r]} q_j - Q_k$. By the assumption of being in the L-costly-event case, this decrease is at most γ times LWF's original wait of $\left(q_0 + \sum_{j\in[1,r]} (\ell_j p_j + q_j)\right)$.

We now argue that competitive ratio that the adversary achieves in the new game is at least c. 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. In the creation of an A-costly-event, it is obvious that LWF's wait time does not change, and we need to argue that the adversary's wait time goes up by at most a factor of $\frac{1}{\gamma}$. The adversary's wait time increases to the wait time for LWF. Since this is an A-costly-event, LWF's wait time is at most $\frac{1}{\gamma}\left(\sum_{j\in[0,r]} q_j - Q_k\right)$, which is at most $\frac{1}{\gamma}\sum_{j\in[0,r]} q_j$. The adversary's wait time in the old game was $\sum_{j\in[0,r]} q_j$. $\square$

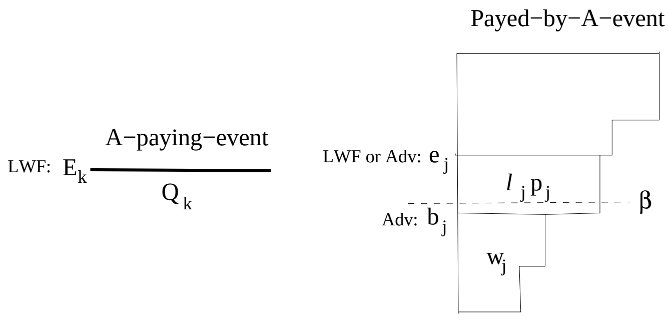

Figure 3: An A-paying-event paying a payed-by-A-event

For the rest of this section we analyze LWF under the rules of the new game. We will now define various types of events that are used in the analysis. Each A-costly-event can 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 the time that this event occurs by E_k, which is the time that all its requests are serviced by LWF. We use Q_k to denote the collection of requests serviced by this event. 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 $\text{OPT}_1 \geq \sum_{k\in\mathcal{A}} Q_k$.

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. Let Q_j be the requests serviced by the event j. 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

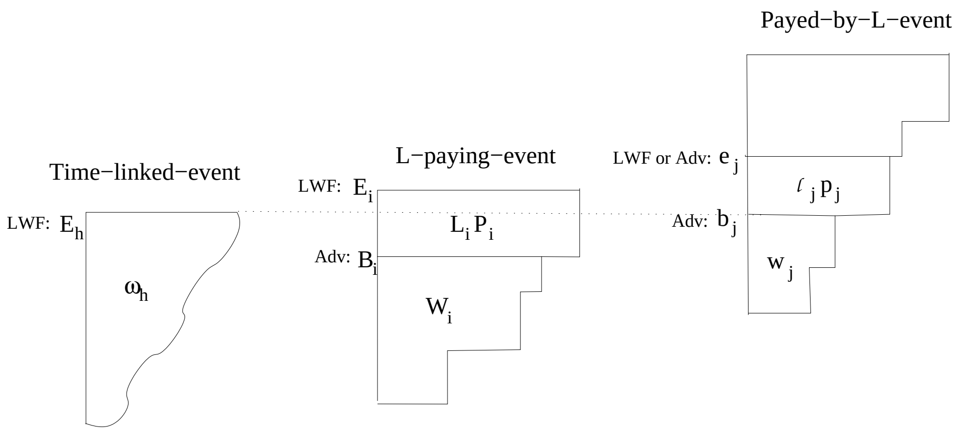

Figure 4: A time-linked-events and an L-paying-event paying a payed-by-L-event

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 before, let $\ell_j = e_j - b_j$ 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 remain unserviced by LWF. Because the input is canonical, and j is an L-costly event, no new requests for this page arrive during $[b_j, e_j - 2]$. Hence, $\ell_j p_j$ is the wait time for this page that accumulates for LWF during this period. It follows that $\text{LWF}_s = \sum_{j\in\mathcal{L}} \ell_j p_j + \sum_{j\in\mathcal{L}\cup\mathcal{A}} Q_j$, or equivalently $\text{LWF}_s - \sum_{j\in\mathcal{L}\cup\mathcal{A}} Q_j = \sum_{j\in\mathcal{L}} \ell_j p_j$. Then since $\text{OPT}_1 \geq \sum_{j\in\mathcal{L}\cup\mathcal{A}} Q_j$ it follows that

$$\text{LWF}_s - \text{OPT}_1 \leq \sum_{j\in\mathcal{L}} \ell_j p_j$$

We partition as follows the set $\mathcal{L}$ of payed-events into the set $\mathcal{L}_A$ of *payed-by-A-events* and the set $\mathcal{L}_A$ of *payed-by-L-events* depending on whether it is to be payed for by A-paying-events or L-costly-events. The thresholds for this partitioning are β and ϵ which are be some small constants to be defined later. During the last $1-\beta$ fraction of event j's life, namely $[b_j + \beta\ell_j, e_j]$, LWF services a total of $(1-\beta)s\ell_j$ times. We say these servicings are *late for* j. 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, it is a payed-by-L-event.

An L-costly-event of the form $[b_r, E_i]$, 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, $L_i = E_i - B_i$, $W_i = w_r$, and $P_i = p_r$. In total, the wait time of these requests under LWF is $W_i + L_i P_i$ is $\sum_{i\in\widehat{\mathcal{L}}} L_i P_i \leq \text{LWF}_s$.

In addition to many L-paying-events, each payed-by-L-event is payed by what we call time-linked-events. See Figure 4. A *time-linked-event* can either be A or L-costly. There is one time-linked-event for each time LWF services a page, and we will index the time-linked-events 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 μ_h denote the total wait time for LWF for this page at the time E_h. It follows that $\sum_{h\in\mathcal{H}} \mu_h = \text{LWF}_s$. There are s time-linked-events h ending and at most one payed-by-L-event i beginning during any one time step $E_h = b_j$. There is at most one payed-by-L-event at this time because the adversary has a unit speed processor. These time-linked-events will pay this payed-by-L-event. Among the s time linked events, let h_j be the one with the smallest wait time at time E_h.

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

Lemma 3

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

Lemma 4

$$\sum_{h\in\mathcal{H}} \mu_h + \sum_{i\in\widehat{\mathcal{L}}} L_i P_i \geq (1-\gamma)^2 \cdot \frac{((1-\epsilon-\beta)s-1)^2}{2s}\left(\sum_{j\in\mathcal{L}_L} \ell_j p_j - 2Q\right)$$

Proof: *(Theorem 1)* We multiply the equation in Lemma 3 by $(1-\gamma)\cdot\frac{((1-\epsilon)s-1)^2}{2s}$ and add it to the equation in Lemma 4 to get

$$\begin{aligned}
&\sum_{h\in\mathcal{H}} \mu_h + \sum_{i\in\widehat{\mathcal{L}}} L_i P_i + \left(\frac{(s+1)}{\epsilon\beta s}\right)\left((1-\gamma)\cdot\frac{((1-\epsilon)s-1)^2}{2s}\right)\sum_{k\in\mathcal{A}} Q_k \\
\geq\;& (1-\gamma)^2\cdot\frac{((1-\epsilon-\beta)s-1)^2}{2s}\left(\sum_{j\in\mathcal{L}_A}\ell_j p_j + \sum_{j\in\mathcal{L}_L}\ell_j p_j - 2Q\right) \\
=\;& (1-\gamma)^2\cdot\frac{((1-\epsilon-\beta)s-1)^2}{2s}\left(\sum_{j\in\mathcal{L}}\ell_j p_j - 2Q\right)
\end{aligned}$$

The equality in the equation above holds since $\mathcal{L}_A \cup \mathcal{L}_L = \mathcal{L}$. Let $\alpha' = \left(\frac{(s+1)}{\epsilon\beta s}\right)\left((1-\gamma)\cdot\frac{((1-\epsilon)s-1)^2}{2s}\right)$ and let $\alpha = \left((1-\gamma)^2\cdot\frac{((1-\epsilon-\beta)s-1)^2}{2s}\right)$. Substituting the inequalities $\sum_{h\in\mathcal{H}}\mu_h = \text{LWF}_s$, $\sum_{i\in\widehat{\mathcal{L}}} L_i P_i \leq \text{LWF}_s$, $\sum_{k\in\mathcal{A}} Q_k \leq \text{OPT}_1$, $\sum_{j\in\mathcal{L}_A}\ell_j p_j \geq \text{LWF}_s - \text{OPT}_1$, and $Q \leq \text{OPT}_1$ yields

$$\text{LWF}_s + \text{LWF}_s + \alpha'\text{OPT}_1 \geq \alpha\left(\text{LWF}_s - \text{OPT}_1 - 2\text{OPT}_1\right)$$

By algebra this is equivalent to

$$(3\alpha + \alpha')\,\text{OPT}_1 \geq (\alpha - 2)\,\text{LWF}_s$$

Now in order to see that LWF is $O(1)$-competitive for the new game, it is sufficient that $\alpha > 2$. By taking ϵ, γ and β to be very small, we need that

$$\frac{(s-1)^2}{2s} > 2$$

Note that this equation holds for $s = 6$. Finally, $O(1)$-competitiveness for the original game follows from Lemma 2. $\square$

The first step toward proving Lemma 3 uses the order that LWF services the pages to obtain an inequality on some of the variables involved. This establishes a payment from an A-paying-event to a payed-by-A-event.

Lemma 5 *Let $k \in \mathcal{A}$ be an A-paying-event and $j \in \mathcal{L}_A$ be an payed-by-A-event such that $b_j < E_k \lesssim e_j$. Then it is the case that*

$$Q_k \geq (1-\gamma)(E_k - b_j)p_j$$

Proof: LWF services the page associated with the A-paying-event k at time E_k, but because $E_k \gtrsim e_j$, 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 at least $w_j + (E_k - b_j)p_j$, 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 $[b_j, E_k]$. The wait time for j at time E_k may be strictly larger than $(E_k - b_j)p_j$ if new requests for this page arrived at time $E_k - 1$. It follows that $Q_k \geq (1-\gamma)\left(w_j + (E_k - b_j)p_j\right) \geq (1-\gamma)(E_k - b_j)p_j$. □

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

Lemma 6 *If $k \in \mathcal{A}$ is an A-paying-event and $j \in \mathcal{L}_A$ is an payed-by-A-event, let $a_{k,j}$ be an associated 0/1 indicator variable. Then there is setting of the variables $a_{k,j}$'s so that:*

- *If $a_{k,j} = 1$ then $b_j < E_k \gtrsim e_j$, and k is late for j.*
- *For all k, it is the case that $\sum_{j \in \mathcal{L}_A} a_{k,j} \leq \frac{s+1}{\epsilon s}$.*
- *For all j, it is the case that $\sum_{k \in \mathcal{A}} a_{k,j} \geq 1$.*

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

Proof: *(Lemma 3)* We use the variables $a_{k,j}$ from Lemma 6 as coefficients in a linear combination of the inequalities from Lemma 5. That is,

$$\sum_{k \in \mathcal{A}, j \in \mathcal{L}_A} a_{k,j} Q_k \geq \sum_{k \in \mathcal{A}, j \in \mathcal{L}_A} a_{k,j} \left((1-\gamma)(E_k - b_j)p_j\right)$$

Having $b_j < E_k \gtrsim e_j$ when $a_{k,j} \neq 0$ ensures that Lemma 6 holds. Having k is late for j ensures that it is during the last $1-\beta$ fraction of event j's life, namely $E_k \in [b_j + \beta \ell_j, e_j]$, giving that $E_k - b_j \geq \beta \ell_j$. Therefore

$$\sum_{k \in \mathcal{A}, j \in \mathcal{L}_A} a_{k,j} Q_k \geq \sum_{k \in \mathcal{A}, j \in \mathcal{L}_A} a_{k,j} \left((1-\gamma)\beta \ell_j p_j\right)$$

By Lemma 6 $\sum_{j \in \mathcal{L}_A} a_{k,j} \leq \frac{s+1}{\epsilon s}$ and $\sum_{k \in \mathcal{A}} a_{k,j} \geq 1$, and hence it must be the case that

$$\sum_{k \in \mathcal{A}} \frac{s+1}{\epsilon s} Q_k \geq \sum_{j \in \mathcal{L}_A} \left((1-\gamma)\beta \ell_j p_j\right)$$

The claim then follows by dividing by β. □

Proof: *(Lemma 6)* 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 and only if $b_j < E_k \gtrsim e_j$, and k is late for 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. So consider a particular fixed S.

For each paid-by-A-event $j \in S$, let I_j be the collection of A-paying-events j such that $b_j < E_k \gtrsim e_j$, and k is late for j. Let $\mathcal{I}$ be the collection of such I_j for $j \in S$. Note that each $I_j \in \mathcal{I}$ consists of an interval of contiguous LWF servicings. We then construct a subcollection $\mathcal{J}$ of $\mathcal{I}$ in the following manner, starting with step 1:

1. If there is an interval $I_j \in \mathcal{I}$ that starts later than the ending time of every interval in $\mathcal{J}$ then we add the earliest starting interval with this property to $\mathcal{J}$, and go to the next step. If no such interval exists then halt.

2. If there is an interval in $\mathcal{I}$ that intersect the previous intervals $\mathcal{J}$ but has a later ending time, then add to $\mathcal{J}$ one such interval with the latest LWF servicing, and repeat this step. Else repeat the previous step.

It is easy to see that $\bigcup \mathcal{J} = \bigcup \mathcal{I}$. Let $\mathcal{J}_o$ contain the first, third, fifth, etc. intervals added to $\mathcal{J}$, and $\mathcal{J}_e$ contain the second, fourth, sixth, etc. intervals added to $\mathcal{J}$. Then it is easy to see that at least one of the following is true: $|\bigcup \mathcal{J}_o| \geq |\bigcup \mathcal{J}|/2$, or $|\bigcup \mathcal{J}_e| \geq |\bigcup \mathcal{J}|/2$. Without loss of generality, assume that $|\bigcup \mathcal{J}_o| \geq |\bigcup \mathcal{J}|/2$.

Now consider an interval $I_j \in \mathcal{J}_o$. Since j is a payed-by-A-event, it contains at least $\epsilon s \ell_j$ A-paying events i that are late for j. Hence our selected intervals cover at least $\epsilon s |\bigcup \mathcal{I}|/2$ 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 $N(S) \geq \frac{s+1}{\epsilon s} \cdot (\epsilon s |\bigcup \mathcal{I}|) = (s+1)|\bigcup \mathcal{I}|$. Note that $|S| \leq (s+1)|\bigcup \mathcal{I}|$ since for each LWF servicing $i \in \mathcal{I}$ there are at most s LWF servicings j at this time, and at most one adversarial servicing at this time. Combining $|S| \leq (s+1)|\bigcup \mathcal{I}|$, and $(s+1)|\bigcup \mathcal{I}| \leq |N(S)|$, gives that $|S| \leq |N(S)|$. This this establishes by Hall's theorem that the desired matching exists. $\square$

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. We recommend referring to figure 4. We will consider a time-linked-event $h \in \mathcal{H}$ that happens at the same time as a payed-by-L-event $j \in \mathcal{L}_L$ that both happen at time $E_h = b_j$. We further consider a L-paying-event $i \in \widehat{\mathcal{L}}$ such that $B_i < b_j = E_h < E_i \lesssim e_j$. We will then use these relationships to establish a payment scheme that will eventually allow us to prove Lemma 4.

Lemma 7 *Let $h \in \mathcal{H}$ be a 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 such that $B_i < b_j = E_h < E_i \lesssim e_j$. Then it must be the case that*

$$\mu_h + (E_i - b_j)P_i \geq (1-\gamma)^2 (E_i - b_j)p_j$$

Proof: At time E_h, the wait time on h is μ_h and the wait time on i is $W_i + (E_h - B_i)P_i$. Since $E_h < E_i$, according to the new rules for LWF, it must be the case that $\mu_h \geq (1-\gamma)\left(W_i + (E_h - B_i)P_i\right)$, or equivalently $\frac{\mu_h}{1-\gamma} - (E_h - B_i)P_i \geq W_i$.

At time E_i, the wait time on i is $W_i + L_i P_i$ and the wait time on j is at least $w_j + (E_i - b_j)p_j$ (it might be more if more requests for the associated page arrived at time $E_i - 1$. Since, $E_i \lesssim e_j$, it must be the case that $W_i + L_i P_i \geq (1-\gamma)\left(w_j + (E_i - b_j)p_j\right)$.

Substituting, we get that $\frac{\mu_h}{1-\gamma} - (E_h - B_i)P_i + L_i P_i \geq (1-\gamma)\left(w_j + (E_i - b_j)p_j\right)$. By dropping the w_j term, we get that $\frac{\mu_h}{1-\gamma} - (E_h - B_i)P_i + L_i P_i \geq (1-\gamma)\left((E_i - b_j)p_j\right)$. We now consider the term $(E_h - B_i)P_i + L_i P_i$ and find that:

$$(E_h - B_i)P_i + L_i P_i = P_i(L_i - (E_h - B_i)) = P_i((E_i - B_i) - (E_h - B_i)) = P_i(E_i - E_h) = P_i(E_i - b_j)$$

Hence, $\frac{\mu_h}{1-\gamma} - (E_i - b_j)P_i \geq (1-\gamma)\left(w_j + (E_i - b_j)p_j\right)$, or equivalently, $\mu_h + (1-\gamma)(E_i - b_j)P_i \geq (1-\gamma)^2(E_i - b_j)p_j$. Finally, by noting that $(1-\gamma) < 1$, we get the desired result that $\mu_h + (E_i - b_j)P_i \geq (1-\gamma)^2(E_i - b_j)p_j$. $\square$

Now Lemma 4 follows by taking a linear combination of the inequalities from Lemma 7.

Proof: *(Lemma 4)* 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:

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

Using this both as a scaling factor for the equations from Lemma 7 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 $b_j = E_h$: $a_{i,j}\left(\mu_h + (E_i - b_j)P_i\right) \geq a_{i,j}\left((1-\gamma)^2(E_i - b_j)p_j\right)$. Since LWF services s pages at time b_j, there are s time-linked-events for which $b_j = E_h$. Recall that h_j is the time linked event with the smallest wait time at time E_h. Summing over all i and j, the above equation gives

$$\sum_{i\in\widehat{\mathcal{L}},\ j\in\mathcal{L}_L} a_{i,j}\mu_{h_j} + \sum_{i\in\widehat{\mathcal{L}},\ j\in\mathcal{L}_L} a_{i,j}(E_i - b_j)P_i \geq (1-\gamma)^2 \sum_{i\in\widehat{\mathcal{L}},\ j\in\mathcal{L}_L} a_{i,j}(E_i - b_j)p_j \tag{1}$$

We now use three Lemmas that we will prove momentarily. Lemma 8 states that $\sum_{i\in\widehat{\mathcal{L}},\ j\in\mathcal{L}_L} a_{i,j}\mu_{h_j} \leq \sum_{h\in\mathcal{H}}\mu_h$. Lemma 9 states that $\sum_{i\in\widehat{\mathcal{L}},\ j\in\mathcal{L}_L} a_{i,j}(E_i - b_j)P_i \leq \sum_{i\in\widehat{\mathcal{L}}} L_i P_i$. Hence, substituting these results into equation 1 give

$$\sum_{h\in\mathcal{H}}\mu_h + \sum_{i\in\widehat{\mathcal{L}}} L_i P_i \geq (1-\gamma)^2 \sum_{i\in\widehat{\mathcal{L}},\ j\in\mathcal{L}_L} a_{i,j}(E_i - b_j)p_j \tag{2}$$

Lemma 10 states that $\sum_{i\in\widehat{\mathcal{L}},\ j\in\mathcal{L}_L} a_{i,j}(E_i - b_j)p_j \geq \frac{((1-\epsilon-\beta)s-1)^2}{2s}\left(\sum_{j\in\mathcal{L}_L}\ell_j p_j - 2Q\right)$. Substituting this result into equation 2 gives

$$\sum_{h\in\mathcal{H}}\mu_h + \sum_{i\in\widehat{\mathcal{L}}} L_i P_i + \geq (1-\gamma)^2 \cdot \frac{((1-\epsilon-\beta)s-1)^2}{2s}\left(\sum_{j\in\mathcal{L}_L}\ell_j p_j - 2Q\right) \tag{3}$$

as we wanted to prove. □

Lemma 8

$$\sum_{i\in\widehat{\mathcal{L}},\ j\in\mathcal{L}_L} a_{i,j}\mu_{h_j} \leq \sum_{h\in\mathcal{H}}\mu_h$$

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 $b_j < E_i \lesssim e_j$, because at each time step 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 the 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 $\mu_{h_{j,1}}, \ldots, \mu_{h_{j,s}}$. By definition of h_j, each of these wait times is at least μ_{h_j}. It then follows that

$$\sum_{h\in\mathcal{H}}\mu_h \geq \sum_{j\in\mathcal{L}_L}(\mu_{h_{j,1}} + \ldots + \mu_{h_{j,s}}) \geq \sum_{j\in\mathcal{L}_L} s\mu_{h_j} \geq \sum_{j\in\mathcal{L}_L}(\sum_{i\in\widehat{\mathcal{L}}} a_{i,j})\mu_{h_j}$$

□

Lemma 9

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

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 $B_i < b_j < E_i$, because at each time step the adversary services at most one page. Therefore the number of j for which $a_{i,j}$ is non-zero is at most L_i. When $a_{i,j}$ is non-zero,

$$a_{i,j}(E_i - b_j) = \frac{1}{\ell_j}(E_i - b_j) = \frac{1}{(e_j - b_j)}(E_i - b_j) \le \frac{1}{(e_j - b_j)}(e_j - b_j) = 1$$

The inequality in the above equation holds since when $a_{i,j}$ is non-zero, it is the case that $E_i \lesssim e_j$. This gives that

$$\sum_{i \in \widehat{\mathcal{L}}} \sum_{j \in \mathcal{L}_L} a_{i,j}(E_i - b_j)P_i \le \sum_{i \in \widehat{\mathcal{L}}} \sum_{j \in \mathcal{L}_L} P_i \le \sum_{i \in \widehat{\mathcal{L}}} L_i \cdot P_i$$

□

Lemma 10

$$\sum_{i \in \widehat{\mathcal{L}},\ j \in \mathcal{L}_L} a_{i,j}(E_i - b_j)p_j \ge \frac{((1-\epsilon-\beta)s-1)^2}{2s} \left(\sum_{j \in \mathcal{L}_L} \ell_j p_j - 2Q \right)$$

Proof: For each $j \in \mathcal{L}_L$, we will show that the following inequality holds:

$$\begin{aligned} \sum_{i \in \widehat{\mathcal{L}}} a_{i,j}(E_i - b_j)p_j &= \frac{p_j}{\ell_j} \sum_{i \in \widehat{\mathcal{L}}, a_{i,j} \neq 0} (E_i - b_j) && (4) \\ &\ge \frac{((1-\epsilon-\beta)s-1)^2}{2s} \left(\ell_j p_j - \begin{cases} 2p_j & \text{if } e_j \text{ is an LWF servicing} \\ 0 & \text{otherwise} \end{cases} \right) && (5) \end{aligned}$$

The proof then follows by summing up over all such j. We break the proof into cases. In the first case let $j \in \mathcal{L}_L$ be any payed-by-L-event which ends with an explicit servicing by the adversary (i.e. subtracting 0 and not p_j in equation 5).

Our first goal is to count the number, denoted by m_j, of $i \in \widehat{\mathcal{L}}$ for which $a_{i,j}$ is non-zero, namely the number of L-paying-events i for which $B_i < b_j < E_i \lesssim e_j$. There are $s(e_j - b_j) = s\ell_j$ LWF servicings i satisfying $b_j < E_i \lesssim e_j$ since LWF has a speed s processor and the adversary servicing j at time E_j happened after all LWF servicings at time E_j. We subtract off of these the number that are A-paying-events. Since j is a payed-by-L-event, at most $\epsilon s\ell_j$ of the LWF servicings i, that are late for j, are A-paying-events. It is possible that all of the early servicings are A-paying-events, however, there are at most $\beta s\ell_j$ of these. This leaves $(1-\epsilon-\beta)s\ell_j$ L-paying-events i with $b_j < E_i \lesssim e_j$. Now we must subtract off the number of these for which $b_j \le B_i$, but there are at most $e_j - b_j = \ell_j$ of these, because the adversary services at most one page at each time step. (Note that $i \neq j$ because i is a L-paying-event that ends with an LWF servicing, and the payed-by-L-event j by assumption ends with an an explicit adversary servicing.) In conclusion, the number m_j of $i \in \widehat{\mathcal{L}}$ for which $a_{i,j}$ is non-zero, is at least $s\ell_j - \epsilon s\ell_j - \beta s\ell_j - \ell_j = ((1-\epsilon-\beta)s-1)\ell_j$.

Our goal is to lower bound the sum $\sum_{i \in \widehat{\mathcal{L}}, a_{i,j} \neq 0}(E_i - b_j)$ In the worst case, each $E_i - b_j$ is as small as it can be. (Because we subtracted off from our count m_j the early LWF servicings, we know that $E_i - b_j \ge \beta\ell_j$. However, this does not significanly improve the bound so we will only use the fact from $b_j < E_i$ that $E_i - b_j \ge 1$.) 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, ..., s for which it is $\lfloor \frac{m_j}{s} \rfloor$, and finally the remainder of them which are $\lfloor \frac{m_j}{s} \rfloor + 1$. Assume briefly, that m_j is divisible by s. Then this gives

$$\sum_{i \in \widehat{\mathcal{L}}, a_{i,j} \neq 0} (E_i - b_j) \geq \sum_{t=1}^{m_j/s} st = s\left(\frac{m_j}{s}\right)\left(\frac{m_j}{s} + 1\right)/2 \geq \frac{m_j^2}{2s}. \quad (6)$$

We will leave it as an exersize that the same bound is true when m_j is not divisible by s.

Let us return our attention to equation 4. Recall that when $a_{i,j}$ is non-zero, it is equal to $\frac{1}{\ell_j}$. Therefore, we know that the left hand side of the inequality in equation 4 is at least

$$\frac{p_j}{\ell_j} \cdot \frac{m_j^2}{2s} = \frac{((1-\epsilon-\beta)s-1)^2}{2s} \ell_j p_j$$

For the second case assume that e_j represents an LWF servicing. Now we can only argue that there are $s(e_j - b_j - 1) = s(\ell_j - 1)$ LWF servicings at an $E_i \in [b_j + 1, e_j]$ with $E_i \lesssim e_j$ since e_j may be the first LWF servicing at time e_j. Thus using the argument in the preceding case, we get that

$$\frac{p_j}{\ell_j} \sum_{i \in \widehat{\mathcal{L}}, a_{i,j} \neq 0} (E_i - b_j) \geq \frac{((1-\epsilon-\beta)s-1)^2}{2s}\left(\frac{(\ell_j - 1)^2}{\ell_j} p_j\right) \geq \frac{((1-\epsilon-\beta)s-1)^2}{2s}(\ell_j p_j - 2p_j) \quad (7)$$

Summing up the p_j over the Payed-by-L-events that end with an LWF servicing is then at most the total number of requests Q. □

3 Lower Bound

Theorem 11 *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 by the definition of s, the n_1 requests in S_1 are to the same pages as the first n_1 requests in S_0. Further, the first $n - n_1$ requests in S_2 are to the same pages as the last $n - n_1$ requests in S_0. The fact that $n < n_1 + n_2$ follows from $s < \frac{1+\sqrt{5}}{2}$. After the requests in S_2, 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 proportional to the number of such requests. LWF first does the requests in S_0 in order, finishing these at time $n_1 = n/s$. 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. We think that it is helpful to keep figure 3 in mind. The horizontal axis are the various pages, and the vertical axis is time. A dot represents requests for that page at that time. The solid line shows that the adversary answers requests in some S_r, $r \geq 1$, as they arrive. The dashed line represents the order that LWF services the requests. As in figure 2, you can find the response time for a query for a particular scheduling algorithm by measuring the vertical distance from the arrival dot to the completion curve for that scheduling algorithm. Thus the total response time for any algorithm is a region under its completion curve. The doubly shaded region represents the total response time that is common to LWF and the adversary for requests in S_0. The singly shaded region represents total response time for LWF that the adversary doesn't experience.

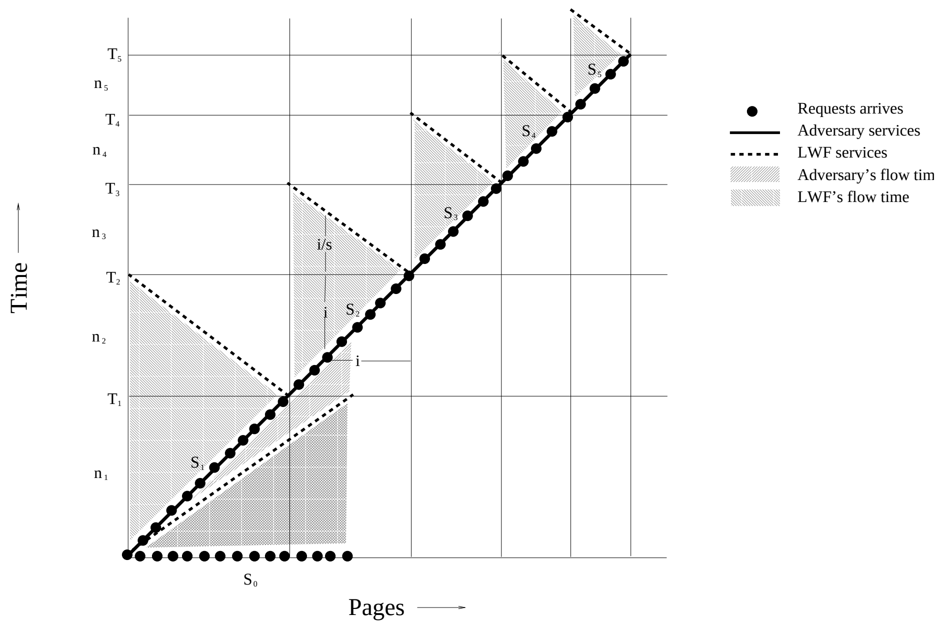

Figure 5: The time that requests for requests arrive and are serviced by LWF and by the adversary.

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 $\left(\frac{i}{s}\right) - (T_1 - j)$ and wait time $\left(\frac{i}{s} - T_1 + j\right) \cdot w_1 \frac{1}{j} = \left(\frac{i}{s} - \frac{n}{s} + j\right) \cdot \frac{n}{s} \frac{1}{j} = -\frac{n-i}{s} \cdot \frac{n}{sj} + \frac{n}{s}$. 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 $0 \le i < j < n_r$. At time $T_r + \frac{i}{s}$, when $R_{r,i}$ is stated to be serviced, its $\frac{w_r}{i}$ requests have age $\left(T_r + \frac{i}{s}\right) - (T_r - i) = \left((\frac{1}{s}+1)i\right)$ and wait time $\left((\frac{1}{s}+1)i\right) \cdot w_r \frac{1}{i} = (\frac{1}{s}+1)w_r$. At this time, $R_{r,j}$'s $\frac{w_r}{j}$ requests have age $\left(T_r + \frac{i}{s}\right) - (T_r - j) = \left(\frac{i}{s} + j\right)$ and wait time $\left(\frac{i}{s}+j\right) \cdot w_r \frac{1}{j} < \left(\frac{j}{s}+j\right) \cdot w_r \frac{1}{j} = (\frac{1}{s}+1)w_r$. 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 $\left(T_r + \frac{i}{s}\right) - (T_{r+1} - j) = \left(\frac{i}{s} - n_{r+1} + j\right) = \left(\frac{i-n_r}{s} + j\right)$ and wait time $\left(\frac{i-n_r}{s}+j\right) \cdot \frac{w_r}{j}(1+\frac{1}{s}) < j \cdot \frac{w_r}{j}(1+\frac{1}{s}) = w_r(1+\frac{1}{s})$. 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 $\mathrm{LWF}_s(I) \ge \Omega(n_R w_R)$.

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 of 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 $\mathrm{OPT}_1(I) \le O(w_R \ln n_R)$.

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.