Scheduling Broadcasts in Wireless Networks

Bala Kalyanasundaram *
Computer Science Department
Georgetown University
kalyan@cs.georgetown.edu

Kirk R. Pruhs *
Computer Science Department
University of Pittsburgh
kirk@cs.pitt.edu

Mahendran Velauthapillai
Computer Science Department
Georgetown University
mahe@cs.georgetown.edu

Abstract

We consider problems involving how to schedule broadcasts in a pulled-based data-dissemination service, such as the DirecPC system, where data requested by the clients is delivered via broadcast. In particular, we consider the case where all the data items are of equal size and preemption is not allowed. We give an offline $O(1)$-speed $O(1)$-approximation algorithm for the problem of minimizing the average response time. We provide worst-case analysis, under various objective functions, of the online algorithms that have appeared in the literature, namely, *Most Requests First*, *First Come First Served*, and *Longest Wait First*.

1 Introduction

1.1 Motivation, Problem Statement and Background

Recently the WWW has led an explosive demand for pulled-based data-dissemination services where multiple clients request information from a source of information. Complementing the demand for such services is the greatly increased bandwidth that is available in satellite, wireless, and cable TV based networks that employ broadcasting to disseminate information. Broadcasting is particularly appropriate for data dissemination applications since broadcasting can exploit commonalities among the requests from the clients to improve service. For example, if multiple clients make identical requests at roughly the same time (imagine the multitude of requests for www.yahoo.com), then the server can satisfy all of these requests in one broadcast. Data broadcasting is used in several

*Supported in part by NSF Grant CCR-9734927 and by AFOSR grant F49620010011.

commercial systems, including the Intel Intercast System [7], the Hughes DirecPC System [5], the Hybrid System [6], and the AirMedia System [2]. For example, in the Hughes DirecPC system (depicted in figure 1) the clients request data items over phone lines, and the data is delivered via satellite broadcast. In this paper we consider the problem of how to schedule such broadcasts to optimize various objective functions, including minimizing the average response time. We provide worst-case analysis for various offline and online algorithms.

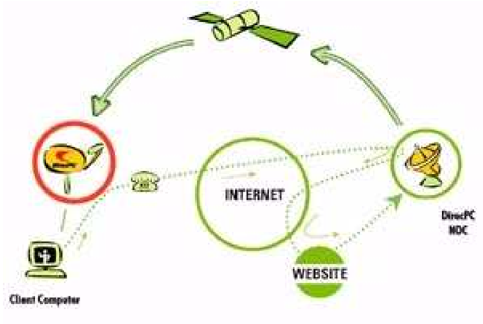

Figure 1: A depiction of the DirecPC System from their WWW site [5])

We formalize the problem in the following way. The setting consists of n possible data items, $P_1, \ldots, P_n$, which we will henceforth call pages. We assume that time is discrete and any page can be broadcasted during one time unit. Suppose we have a k-speed server. We assume that at each time $t \geq 0$, the k-speed server first decides on up to k pages to broadcast at time t, and after the broadcast the server receives $R_i(t)$ new requests for page P_i (for each $1 \leq i \leq n$). Note that, as a consequence, requests that arrive at time t will not be satisfied at time t. We say that the $R_i(t)$ requests for page P_i at time t are *satisfied* at time $C_i(t)$ if $C_i(t)$ is the first time after t when P_i is broadcasted. In the online setting, the server is not aware of the requests $R_i(t)$ until broadcast at time t is complete. In the offline setting, the algorithm is aware of all future requests. Thus in standard scheduling notation [9], the job environment would be $r_i, p_i = 1$.

Consider for the moment the objective function of minimizing the average response time, which is probably the most commonly used measure of system performance [4]. The *response/flow* time of a request for P_i at time t is $(C_i(t) - t)$, which is how long a client that requested page P_i at time t have to wait until P_i is broadcasted. The *average response time* is then

$$\frac{\sum_t \sum_i R_i(t) \cdot (C_i(t) - t)}{\sum_t \sum_i R_i(t)}$$

This represents the average time to respond to a request of a client. Now consider the example shown in table 1. For this instance, the schedule that minimizes the total response time for a unit speed server is to broadcast P_a at time 1, P_b at time 2, and P_c at time 3. This schedule produces a total response time of 14, and an average response time of $\frac{14}{8}$.

Before stating our results, let us review resource augmentation analysis, which was first proposed in [8] as a method for analyzing scheduling algorithms. Generally speaking, resource augmentation analysis compares the performance of a limited algorithm A (e.g. an online algorithm, or polynomial-time algorithm) with more resources (e.g. faster processors or more processors) to an omniscient algorithm with less resources. How this is exactly formalized varies from problem to problem. For

Input			
$R_i(t)$	$i=a$	$i=b$	$i=c$
$t=0$	2	3	1
$t=1$	0	1	1

$R_i(t)(C_i(t)-t)$	$i=a$	$i=b$	$i=c$
$t=0$	2	6	3
$t=1$	0	1	2

Table 1: Example Input, and Total Response Time for the schedule P_a, P_b, P_c

concreteness, let us explain how it is formalized for our broadcast problem with the objective function of minimizing the average response time using the terminology proposed in [11]. An algorithm A is said to be an *s-speed c-approximation* algorithm if

$$\max_{\text{Inputs } I} \frac{A_s(I)}{\textsc{Opt}(I)} \leq c$$

where $A_s(I)$ is the average response time in the schedule produced by A, equipped with a speed s server, on input I, and $\textsc{Opt}(I)$ is value of the objective function in the optimal solution with a unit speed server. An algorithm A is said to be *c-competitive*, or said to have *competitive ratio c*, if A is a 1-speed c-approximation algorithm.

1.2 Related Results

The problem of minimizing the maximum response time was considered in [3] with the job environment $r_i, pmtn$, that is, pages may have variable size (a page of size k would have to be broadcast for k time units to be completed) and preemption is allowed (also it is assumed that the server always broadcasts pages in cyclical order). They give an offline polynomial time algorithm that is $(1+\epsilon)$-competitive. The key insight is that if the optimal maximum response time Opt was known, then this establishes a deadline for every request, and one can easily write an interval-indexed linear program whose solution gives a schedule with maximum response time Opt. One can then obtain a $(1+\epsilon)$-competitive algorithm by performing a binary search on the possible values of Opt. They also showed that First Come First Served (FCFS) is 2-competitive, and that no deterministic online algorithm can be better than 2-competitive.

There have also been several empirical, or average case, studies of broadcast scheduling problems, see for example [1], which also provides a nice introduction to broadcast scheduling.

1.3 Our Results

To date there have been no worst-case results on how to schedule broadcasts to minimize the average response time, which is perhaps the most common measure of system performance [4].

In section 2 we consider the offline version of the problem of minimizing the average response time. We give an offline $O(1)$-speed $O(1)$-approximation polynomial-time algorithm. The algorithmic techniques for the problem of minimizing maximum response from [3] do not apply to average response time because the knowledge of the optimal average response time does give deadlines for individual jobs. We surmount this difficulty by showing how to express the problem of minimizing the average response time as a integer linear program. The optimal solution LOpt of the relaxed fractional linear program is then a lower bound for Opt. From LOpt we can determine a deadline for each request that, if met by every request, would essentially guarantee that every request has response time of $O(1)$ times its response time in OPT. We then show how to meet these deadlines using a variation of Earliest Deadline First scheduling.

In section 3 we consider online broadcast scheduling. We particularly want to analyze the algorithms that have been proposed in the literature (see for example [1]) that are appropriate for our setting, namely:

- Most Requests First (MRF): Broadcast the pages that have the most unsatisfied requests.
- First Come First Served (FCFS): Broadcast the pages with the earliest unsatisfied requests.
- Longest Wait First (LWF): Broadcast the pages with maximum total waiting time. The total waiting time for a page is the total time that all unsatisfied requests for that page are waiting.

In subsection 3.1, we show that there is no deterministic online algorithm that is $O(1)$-competitive with respect to average response time. We then turn our attention to $O(1)$-speed $O(1)$-approximation online algorithms. We show that neither MRF nor FCFS are $O(1)$-speed $O(1)$-approximation online algorithms with respect to average response time. We conjecture that LWF is in fact an $O(1)$-speed $O(1)$-approximation algorithm with respect to average response time. While we do not provide a proof to this conjecture, we show that this cannot be proved in the standard way. Let $U_A^s(t)$ be the number of requests unsatisfied by algorithm A at time t using a speed s server for some generic input. To show that an algorithm A is an s-speed c-approximation algorithm, with respect to average response time, it is sufficient to show that $\max_t U_A^s(t)/U_{\text{OPT}}^1(t) \leq c$ [4]. In fact, this is the most commonly used technique to show that an algorithm A is an c-approximation algorithm with respect to average response time (e.g. [4, 10, 8]). We show, in subsection 3.2, that there is no deterministic online algorithm A that can bound $\max_t U_A^s(t)/U_{\text{OPT}}^1(t)$ by any constant. Thus, for some inputs, there must be times when any algorithm, equipped with an $O(1)$-speed server, will have a factor of $\omega(1)$ more unsatisfied requests than the adversary. And thus, one can not prove that an online algorithm is an $O(1)$-speed $O(1)$-approximation algorithm with respect to average response time in this standard way.

While FCFS and MRF may perform badly with respect to average response time, we show in the subsection 3.3 and subsection 3.4 that there are other natural objective functions under which MRF is superior to FCFS and LWF, and under which FCFS is superior to MRF and LWF.

We show in subsection 3.3 that while an $O(1)$-speed online algorithm can not be competitive with respect to the number of unsatisfied requests, FCFS is an $O(1)$-speed $O(1)$-approximation algorithm with respect to minimizing the number of pages with unsatisfied requests. More precisely, let $\tilde{U}_A^s(t)$ be the number of pages that have unsatisfied requests at time t using algorithm A and an s speed server for some generic input. Observe that that it may well be the case that $U_A^s(t)$ is much greater than $U_A^s(t)$ if multiple clients request common pages. Consider the objective function of minimizing $\max_t \tilde{U}_A^s(t)/\tilde{U}_{\text{OPT}}^1(t)$. We show that FCFS is a 4-speed 4-approximation algorithm for this objective function. That is, FCFS using a 4-speed server guarantees that at every time the number of pages with unsatisfied requests is at most 4 times the minimum possible number of pages with unsatisfied requests that is achievable by a unit speed server. We show that this is in some sense tight by showing the FCFS is not a 3-speed $O(1)$-approximation algorithm for this objective function. It is easy to see that MRF and LWF are not $O(1)$-speed $O(1)$-approximation algorithms for this objective function.

In subsection 3.4 we consider the problem of maximizing the number of satisfied requests. Let $S_A^s(t)$ be the number of requests satisfied by algorithm A by time t using an s speed server on some generic input. We now consider the objective function of minimizing $max_t S^1_{\text{OPT}}(t)/S^1_A(t)$. We show that MRF is 2-competitive with respect to this objective function, that is, at all times, MRF has satisfied at least half as many requests as the maximum possible number of requests satisfied by this time. It is easy to see that neither FCFS nor LWF are $O(1)$-competitive with respect to this objective function.

Thus there are reasonable objective functions under which each of the three algorithms proposed in the literature seems better than the other two.

2 Offline Approximation

In this section we give a 3-speed 3-approximation polynomial-time algorithm that produces a schedule LEDF. We start by reducing the broadcast problem to an integer program IP. The program IP can be best be thought of as a type of min-cost max-flow problem in a directed graph $G = (V, E)$, with side constraints on the capacities of collections of vertices.

Let T be the latest time that a page was requested. The vertex set V contains one vertex $v_i(t)$ for each page P_i, $1 \le i \le n$, and each time t, $0 \le t \le T + n$. In addition, the set V also contains a source vertex s, and a destination vertex d. For the ease of notation, we use $v_i(-1)$ to denote s and $v_i(T + n + 1)$ to denote d, $1 \le i \le n$. Recall by our time conventions that the first requests arrive at time 0, and that at any time t it is only possible for the server to satisfy requests that arrived strictly before time t.

There is a directed edge $e_i(-1, 0)$ in E from s to each $v_i(0)$, $1 \le i \le n$, with capacity 1 and cost 0. For each $0 \le t < t' \le T + n$, and for each $1 \le i \le n$, there is a directed edge $e_i(t, t')$ in E from $v_i(t)$ to $v_i(t')$ with capacity 1, and cost $w_i(t, t') = \sum_{j=t}^{t'-1} (t' - j) R_i(j)$; A flow of 1 on this edge corresponds to broadcasting page P_i at time t, and then next broadcasting page P_i at time t', and thus, $w_i(t, t')$ is the contribution to the total flow time for requests to page P_i that arrive between t and $t' - 1$, inclusive. For each $T + 1 \le t \le T + n$, and for each $1 \le i \le n$, there is a directed edge $e_i(t, T + n + 1)$ in E from $v_i(t)$ to d with capacity 1, and cost 0. The left picture in figure 2 shows the portion of the flow graph G consisting of s, d, the vertices of the form $v_b(*)$ for the input shown in table 1, where the edge labels are the costs.

Call a schedule *k-feasible* if it broadcasts at most k pages at any one time. Note that the reduction of the broadcast problem to the min-cost max-flow is not complete since a flow may not correspond to a 1-feasible broadcast schedule. But it is easy express 1-feasibility as collection a linear constraints. Thus we get the following integer program IP:

$$\min \sum_{t=0}^{T+n} \sum_{t'=t+1}^{T+n} \sum_{i=1}^{n} w_i(t,t') f_i(t,t') \tag{1}$$

$$\sum_{i=1}^{n} f_i(-1,0) = n \tag{2}$$

$$\sum_{t'<t} f_i(t',t) = \sum_{t'>t} f_i(t,t') \qquad 1 \le i \le n \qquad 0 \le t \le T+n \tag{3}$$

$$\sum_{t'<t} \sum_{i=1}^{n} f_i(t',t) \le 1 \qquad 1 \le t \le T+n \tag{4}$$

$$f_i(t,t') \in \{0,1\} \qquad 1 \le i \le n \qquad 0 \le t < t' \le T+n \tag{5}$$

$$f_i(t,n+T+1) \in \{0,1\} \qquad 1 \le i \le n \qquad T+1 \le t \le T+n \tag{6}$$

Constraint 2 guarantees that the total flow is n. Constraint 3 guarantees that flow is conserved at each vertex. Constraint 4 guarantees that the schedule is 1-feasible. The right picture in figure 2 shows a (non-optimal) feasible solution corresponding to the MRF schedule b, a, c, b for the example in table 1. The total response time for this feasible solution is 15.

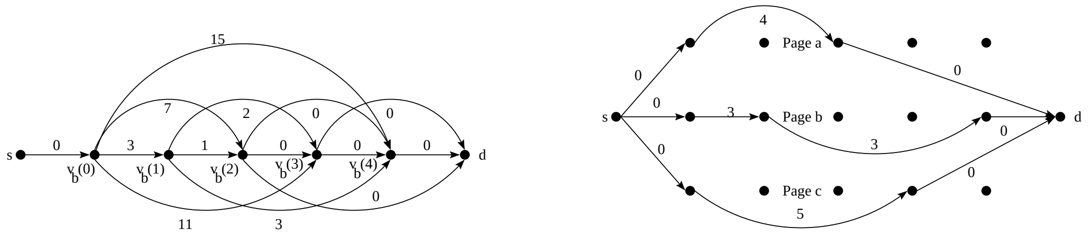

Figure 2: The flow graph

Lemma 1 The optimal value of the objective function in IP is the minimum total response time for the broadcast problem.

Proof: A feasible solution to IP can be converted to a broadcast schedule by broadcasting P_i at time $t > 0$ if and only if there exists a positive flow out of $v_i(t)$. One can convert a feasible solution to the broadcast problem, to a feasible solution to IP by routing the flow through vertices of the form $v_i(t)$ where P_i is broadcast at time t. If a page P_i is last broadcast at a time $t \le T$, the last vertex in the flow through the page P_i can be routed through an arbitrary vertex $v_i(k)$, $T+1 \le k \le T+n$ with the property that no other page is broadcast at time k (such a k must exist since there are n times between $T+1$ and $T+n$, inclusive). ■

We relax IP to get a linear program LP in the standard way by allowing the binary flow variables to take on values in the range $[0,1]$, that is, we replace constraints 5 and 6 by the following constraints:

$$0 \leq f_i(t,t') \leq 1 \qquad 1 \leq i \leq n \quad 0 \leq t < t' \leq T+n \tag{7}$$

$$0 \leq f_i(t,T+n+1) \leq 1 \qquad 1 \leq i \leq n \quad T+1 \leq t \leq T+n \tag{8}$$

Let the optimal solution to LP be denoted by LOPT. We also use LOPT to denote the value of the objective function in this optimal solution to LP. It is obvious that LOPT $\leq$ OPT, and hence to approximate OPT well, it is sufficient to approximate LOPT well.

We now interpret the solution to LP in the following way. Let $p_i(t) = \sum_{t'<t} f_i(t',t)$ be the flow into each vertex $v_i(t)$, for $1 \leq i \leq n$, and for $1 \leq t \leq T+n$. The value $p_i(t)$ can be thought of as the probability that page P_i broadcast at time t. We define $n_i(t)$ to be the earliest time t', such that $t' > t$ and such that $\sum_{j=t+1}^{t'} p_i(t) \geq 1$. Intuitively, $n_i(t)$ is the latest possible time that the requests $R_i(t)$ are satisfied in LOPT. Also, let $\ell_i(t) = 1 - \sum_{j=t}^{n_i(t)-1} p_i(t)$. Intuitively, $\ell_i(t)$ is the probability that the requests $R_i(t)$ are satisfied at time $n_i(t)$ in LOPT.

Now consider the following probabilistic n-feasible solution VL. The $R_i(t) > 0$ requests, for page P_i that arrive at time t, are satisfied at time s, for $t+1 \leq s \leq n_i(t)-1$, with probability $p_i(s)$, and are satisfied at time $n_i(t)$ with probability $\ell_i(t)$. The expected total response time of VL is then:

$$VL = \sum_{i=1}^{n}\sum_{t=1}^{T} R_i(t)\left[\sum_{t'=t+1}^{n_i(t)-1} [p_i(t')\cdot(t'-t)] + \cdot\ell_i(t)(n_i(t)-t)\right] \tag{9}$$

We now show that the expected total response time of the probabilistic solution VL is essentially the same as the one we directly obtained from LOPT.

Lemma 2 $VL =$ LOPT

Proof: We first show that $VL \leq$ LOPT. Consider an arbitrary request $R_i(t)$. Consider the $g(i)$ paths $W_i(1), \ldots, W_i(g(i))$ through vertex $v_i(t)$ in the flow given by LOPT. Assume that these paths are numbered in non-decreasing order of the time that $R_i(t)$ is satisfied in the path. It then follows, by induction on b, that the probability that $R_i(t)$ is satisfied by time $t+b$ in LOPT, $b \geq 1$, is at least $p_i(t+1) + p_i(t+2) + \ldots + p_i(t+b)$. The fact that $VL \leq$ LOPT then follows.

To see that $VL \geq$ LOPT it is sufficient to note that VL may be thought of as a feasible solution to LOPT. ∎

The following lemma states the flow crossing each vertex $v_i(t)$, with $R_i(t) > 0$, in LOPT is at least 1. This can easily be seen by observing the structure of G.

Lemma 3 Suppose $1 \leq i \leq n$, and $1 \leq t \leq T$. If $R_i(t) > 0$ then

$$\sum_{0 \leq j \leq t < k \leq T+n} f_i(j,k) = 1$$

Let $\epsilon < \frac{1}{2}$ be some constant such that $1/\epsilon$ is an integer. Define the jth *break point* of page P_i, denoted $b_i(j)$, to be the earliest time t such that $\sum_{k=1}^{t} p_i(k) \geq j\epsilon$. Let $h(i)$ be the number of break points for page P_i. Let $I_i(j) = [b_i(j), b_i(j+1)]$. We now consider the collection $\mathcal{I}$ of intervals that contain consecutive breakpoints of the same page, that is $\mathcal{I} = \{I_i(j) \mid 1 \leq i \leq n, 1 \leq j \leq h(i)\}$. If a page P_i is broadcast at time $t \in I_i(j)$, then we say that interval I is *satisfied* at time t. We now explain how to create the schedule LEDF.

Algorithm LEDF: Assume that we have scheduled the broadcasts before time t, and now wish to schedule the broadcasts in LEDF at time t. Among the currently unsatisfied intervals that contain time t, select up to $1/\epsilon$ intervals with smallest right endpoint. Broadcast the pages, corresponding to these intervals, at time t.

Lemma 4 Every interval is satisfied in the schedule LEDF.

Proof: The proof is by minimal counter-example. Consider a minimal cardinality instance of k intervals for which the claim is not true. Without loss of generality, let the smallest left endpoint of these intervals be 0, and the smallest right endpoint of an unsatisfied interval be z. No intervals have left endpoints to the right of z by the minimality of the instance.

By construction of the intervals, $\sum_{i=1}^{n}\sum_{t=0}^{z} p_i(t) = \sum_{t=0}^{z}\sum_{i=1}^{n} p_i(t) \geq k\epsilon$. Since the optimal solution to LP, namely LOPT, is 1-feasible, $z \geq k \geq k\epsilon$. Since LEDF employs a $1/\epsilon$ speed server, LEDF could potentially satisfy up to $\frac{z}{\epsilon} \geq k$ intervals during the time interval $[0, z]$. But since not all k intervals were satisfied, it must be the case that LEDF did not use $1/\epsilon$ broadcasts at some time $m \in [0, z)$. Note that by the definition of z, LEDF broadcasted $1/\epsilon$ pages at time z. By the definition of z, it must be the case that every interval with left endpoint in $[0, m]$ must be satisfied by time m. However, then LEDF would fail on the subset of intervals with left endpoints strictly greater than m, which is a strict subset of all of the intervals since at least one interval starts at time 0. We thus reach a contradiction to the minimality of the counter-example. ∎

Lemma 5 The total response time of the $\frac{1}{\epsilon}$-feasible schedule LEDF is at most $\frac{\text{LOPT}}{1-2\epsilon}$.

Proof: Consider an arbitrary page P_i, and an arbitrary time t, such that $R_i(t) > 0$. Consider the earliest break point $b_i(j)$ such that $t < b_i(j)$. One can see that such an interval always exists by lemma 3. Therefore, $t \in [b_i(j-1), b_i(j)]$. By lemma 4, there must exist a time $t' \in [b_i(j), b_i(j+1)]$ such that page P_i is broadcasted at time t'.

By lemma 2, the requests $R_i(t)$ must be satisfied in LOPT at time $b_{j+1}(i)$, or later, with probability at least $1-2\epsilon$. Hence, the contribution to the total response time made by $R_i(t)$ in LOPT is at least $(1-2\epsilon)R_i(t)(b_i(j+1)-t)$.

By lemma 4, $R_i(j)$ is satisfied in LEDF by time $b_{j+1}(i)$. Hence, the contribution to the total response time made by $R_i(t)$ in LEDF is at most $R_i(t)(b_i(j+1)-t)$. The result then follows. ∎

By setting $\epsilon = \frac{1}{3}$, we get the following Theorem.

Theorem 6 There is polynomial-time 3-speed 3-approximation algorithm for the broadcast problem where the objective function is to minimize the average response time.

3 Online Approximation

3.1 Minimizing the Average Response Time

We now show that no online deterministic algorithm with a unit speed server can be constant competitive with respect to minimizing the average response time.

Lemma 7 The competitive ratio of every online deterministic algorithm A, with respect minimizing the average response time, is $\Omega(n)$, where n is the number of distinct pages.

Proof: Assume that n is even. At time 0, every page is requested once. Then no pages are requested until time $n/2$. From time 1 until time $n/2$ the adversary broadcasts the $n/2$ pages not broadcasted by A by time $n/2$. At time $n/2$, the adversary requests all of the pages previously broadcasted by A. Note that there are at most $n/2$ such pages, and that they were not previously broadcasted by the adversary. No more pages are requested until time n.

After the broadcast at time n, the adversary has satisfied all of the requests to date, while A has $n/2$ unsatisfied requests. At each time t, for t from n to $k = n^2$, the adversary requests the page broadcasted by A at time $t-1$. Hence, at each time in $[n, k]$, A had $n/2+1$ unsatisfied requests. At each time $t \in [n+1, k+1]$, the adversary can satisfy the request at time $t-1$. Hence, the adversary has at most 1 unsatisfied request at each time $t \in [n+1, k]$.

We now compute the total response time for the adversary. To satisfy the first n pages, the adversary incurs a cost of $\sum_{i=1}^{n} i = \Theta(n^2)$. Similarly the cost for servicing the next $n/2$ is requests is $\sum_{i=1}^{n/2} i = \Theta(n^2)$. The cost for servicing the last $k-n$ requests is $k-n$. Hence the total response time for the adversary is $O(n^2+k)$.

We now compute the total response time for A. Note that for each time $t \in [n+1, k]$, A has at least $n/2$ unsatisfied requests. Hence, at each such t, the response time of $n/2$ requests increases by one. Hence, the total response time for A is $\Omega(nk)$. The claim then follows by our choice of $k = n^2$. ■

We now show that a faster server does not benefit either MRF or FCFS.

Lemma 8 The algorithm MRF is not an $O(1)$-speed $O(1)$-approximation algorithm with respect to average response time.

Proof: Assume that MRF has an $s = O(1)$ speed processor. Let $k = n^2$. At time 0, the adversary requests pages $P_1, \cdots, P_{n-s}$ once each, and requests pages $P_{n-s+1}, \cdots, P_n$ twice each. At time 1, the algorithm MRF will broadcast the pages $P_{n-s+1}, \cdots, P_n$. At time 1, the adversary broadcasts P_1. At each time t, $1 \le t \le k$, the adversary requests pages $P_{n-s+1}, \cdots, P_n$ twice each. Note that for all times $t \in [1, k]$, MRF will broadcast pages $P_{n-s+1}, \cdots, P_n$. The adversary sequentially broadcasts pages $P_2, \cdots, P_{n-s}$ from time 2 to time $n-s$, and then broadcasts pages $P_{n-s+1}, \cdots, P_n$ in a round robin fashion from time $(n-s)+1$ to time k.

Since the initial requests to pages $P_1, \ldots, P_{n-s}$ are not satisfied by MRF during the first k time units, the total response time for MRF is $\Omega(nk)$, which is $\Omega(n^3)$ since $k = n^2$.

Now we will compute the total response time for the adversary. Each of the $O(ns)$ requests made before time $n-s$ is satisfied within n time units, and each of the $O(ks)$ requests made after time $n-s$ is satisfied within s time units. Hence, the total response time for the adversary is $O(n^2+ks)$, which is $O(n^2)$ since $k = n^2$. Therefore, the approximation ratio for MRF is then $\Omega(n)$. ■

Lemma 9 The algorithm FCFS is not an $O(1)$-speed $O(1)$-approximation algorithm with respect to average response time.

Proof: Let s be the speed of the server used by FCFS. At time 0, the adversary requests pages $P_1, \cdots, P_{n-1}$ once each. At time 1, the adversary makes $s \cdot n^2$ requests for page P_n. The algorithm

FCFS will not broadcast P_n before time n/s. Hence, the total response time for FCFS is $\Omega(n^3)$. The adversary broadcasts page P_n first, and then broadcasts the rest of the pages. The cost to the adversary for P_n is $O(n^2)$, and the cost to the adversary for the rest of the pages is $O(n^2) =$ also. Hence, the approximation ratio is $\Omega(n)$. ■

3.2 Minimizing the Number of Unsatisfied Requests

While we conjecture that LWF is an $O(1)$-speed $O(1)$-approximation algorithm, we now show that this can not be proved in the standard way. That is, no deterministic algorithm A, equipped with an $O(1)$ speed server, can guarantee that, at every time t, the number of requests unsatisfied by A at time t is $O(1)$ times $=$ the number of requests unsatisfied by the adversary at time t. Recall that $U_A^s(t)$ is the number of requests unsatisfied by algorithm A at time s using a s-speed server.

Lemma 10 For all online algorithms A, and for all speeds s, there exists an input such that $\max_t[U_A^s(t)/U_{Adv}^1(t)] = \omega(1)$.

Proof: Let $n = \sum_{i=0}^{s}(s+1)^i$. The n pages form $s+1$ disjoint groups, with $(s+1)^i$ pages in group G_i, $0 \leq i \leq s$. At time 0, each page in G_i is requested n^{s-i} times. Until time b (which we specify below), if A broadcasts a page from the group G_i at time $t \geq 1$, then the adversary requests this page n^{s-i} times at time t.

The adversary maintains a delayed version of A's schedule to determine b. Let B be the delayed version of A's schedule that the adversary maintains. The goal of B is to make sure that the number of broadcasts per group G_i is at most one. Intuitively, suppose schedule A broadcasts multiple pages in G_i at time t, then the schedule B delays the broadcast of all but one page at time t. In order to accomplish this, the adversary maintains a time-dependent set H_i of delayed pages from G_i. Each set H_i is initially empty. If A broadcasts $k \geq 1$ pages from a group G_i at time t, then the schedule B arbitrarily picks one of these k pages to broadcast at time t; The other $k-1$ pages are added to H_i. If A does not broadcast a page from a G_i at time t, and H_i is not empty at time t, then the schedule B arbitrarily selects a page $P_j \in H_i$ and broadcasts P_j at time t. The adversary terminates the construction when there is a group G_i, and a time $b \geq (s+1)^i$, such that B does not broadcast a page from G_i at time b.

Assume to reach a contradiction that the adversary did not terminate by time $b = 2(s+1)^{2s}$. Then for each G_i, B must broadcast at least one page from G_i at each time from $(s+1)^i$ to b. Hence, for each group G_i, there must be $2(s+1)^{2s} - (s+1)^{s+1}$ broadcasts of pages from G_i by time b in the schedule B. Since there are $(s+1)$ groups, B must make $(s+1)[2(s+1)^{2s} - (s+1)^{s+1}]$ broadcasts by time b. Observe that the total number of broadcasts by A, on or before time b, is at least the total number of broadcasts in B, on or before time b. Hence, A must broadcast $(s+1)[2(s+1)^{2s} - (s+1)^{s+1}]$ pages by time b. But since A only has a speed s server, A can broadcast at most $2s(s+1)^{2s}$ pages by time b. By algebraic simplification one can see that this figure is less than $(s+1)[2(s+1)^{2s} - (s+1)^{s+1}]$, and we have our contradiction.

Let G_i be the group responsible for the termination of the construction at time b. We now claim that the adversary can guarantee that it has satisfied all of the requests to the pages in group G_i by time b. To see this, consider the time interval $I = [b - (s+1)^i + 1, b]$. First consider the pages in G_i that are broadcast in B during I. For each such page $P_j \in G_i$, find the latest time $t \in I$ such that B broadcasted P_j at time t. The adversary then broadcasts P_j at time $t+1$. Now consider the pages in G_i that are not broadcasted in B during I. These pages are broadcasted by the adversary at arbitrary times in I when no other broadcasts of pages in G_i are scheduled in B.

So at time b the adversary has no unsatisfied requests for pages from G_i, while A has unsatisfied requests for every page in G_i. Starting from time $b+1$, the adversary will sequentially broadcast all of the pages in groups $G_1, \ldots, G_{i-1}$ until time $d = b+\sum_{j=0}^{i-1}(s+1)^j$. Notice that $s\sum_{j=0}^{i-1}(s+1)^j < (s+1)^i$. Therefore, A still has at least one page from G_i with unsatisfied requests at time d. Therefore, A has at least n^{s-i} requests unsatisfied at time d. On the other hand, the adversary has at most $\sum_{j=i+1}^{s} n^{s-j}(s+1)^j = n^s \sum_{j=i+1}^{s}[(s+1)/n]^j$. This is a decreasing geometric sum with ratio $\frac{n}{s+1}$. Hence, the value of this sum is big-Oh of its largest term. So the adversary has only $O(n^{s-i-1})$ requests unsatisfied at time d. Therefore, at time d, the algorithm A has $\Omega(n)$ times as many unsatisfied requests as does the adversary. ■

Note that the above lemma leaves open the possibility that there exists an algorithm A with the property that $\max_t(U_A^s(t)/U_{Adv}^1(t)) = O(1)$ if we restrict the adversary to be one that minimizes the average response time.

3.3 Minimizing the Number of Pages with Unsatisfied Requests

We now show that FCFS is a 4-speed 4-approximation algorithm with respect to minimizing the number of pages with unsatisfied requests. We also note that MRF and LWF are not $O(1)$-speed $O(1)$-approximation algorithms for this objective function. Recall that $\tilde{U}_A^s(t)$ is the number of pages that have unsatisfied requests at time t using algorithm A and an s speed server for some generic input.

Lemma 11 For all inputs, $\max_t(\tilde{U}_{FCFS}^4(t)/\tilde{U}_{\mathrm{OPT}}^1(t)) \leq 4$.

Proof: Without loss of generality, we assume that FCFS does not idle throughout the time interval under consideration. We break the time into a minimum number of intervals $I_1, I_2, \ldots I_m$. The first interval I_1 begins before any requests arrive, and ends just after the last broadcast before some page is broadcast a second time. Interval I_i, $i > 1$, begins when I_1 ends, and ends just after the last broadcast before some page is broadcast an $(i+1)$st time. Let z_i be the time when I_i starts. Let X_i be the set of pages for which the adversary has unsatisfied requests at time z_i. Let Y_i be the set of pages for which FCFS has unsatisfied requests, but for which the adversary has no unsatisfied requests, at time z_i. For any time t in the interval I_i, let $Z_i(t)$ be the set of pages in $P - X_i$ that are requested during the interval $[z_i, t]$. Recall that P is the set of all pages. Finally, let Z_i be the set of pages in $P - X_i$ that are requested during the interval I_i. Note that $\tilde{U}_{\mathrm{OPT}}^1(z_i) = |X_i|$, and $\tilde{U}_{FCFS}^4(z_i) \subseteq |X_i \cup Y_i|$.

To simplify notation we assume that FCFS broadcasts at a uniform rate of one broadcast every $\frac{1}{4}$th time unit, and that the adversary can broadcast $\frac{1}{4}$th of a page, every $\frac{1}{4}$th time unit.

Applying the definition of the interval, the maximum number of pages satisfied by FCFS during any subinterval $[z_i, t]$ of I_i is $(|X_i|+|Y_i|+|Z_i(t)|)$. Therefore, $t - z_i$ is at most $(|X_i|+|Y_i|+|Z_i(t)|)/4$ and the length of the interval I_i is at most $(|X_i|+|Y_i|+|Z_i|)/4$. As a result, the adversary can satisfy at most $(|X_i|+|Y_i|+|Z_i(t)|)/4$ pages during the interval $[z_i, t]$. Consider a page $P_j \in Y_{i+1}$. It must be the case that $P_j \in X_i \cup Y_i \cup Z_i$, and hence was broadcasted by FCFS during I_i. The fact that a request to P_j is unsatisfied by FCFS at time z_{i+1}, means that after FCFS last broadcasted

P_j during I_i, it was the case that another request for P_j arrived; Furthermore, P_j was broadcasted by the adversary after this arrival, but before time z_{i+1}. Hence, $|Y_{i+1}|$ can be at most the number of pages broadcasted by the adversary during I_i. Hence,

$$|Y_{i+1}| \leq (|X_i| + |Y_i| + |Z_i|)/4 \tag{10}$$

The fact that the adversary satisfies at most $(|X_i| + |Y_i| + |Z_i|)/4$ requests during I_i, also allows us to conclude that

$$|X_{i+1}| \geq |X_i| + |Z_i| - (|X_i + |Y_i| + |Z_i|)/4 \tag{11}$$

We now wish to show by induction on i that $|Y_i| \leq |X_i|$. The base case $i = 0$ ($z_0 = 0$, $|Y_0| = 0$, and $|X_0| = 0$) is obvious. Suppose, $|Y_i| \leq |X_i|$, we will now prove that $|Y_{i+1}| \leq |X_{i+1}|$.

$$\begin{aligned} |X_{i+1}| &\geq \frac{3}{4}|X_i| + \frac{3}{4}|Z_i| - \frac{1}{4}|Y_i| \\ &\geq \frac{3}{4}|X_i| + \frac{1}{4}|Z_i| - \frac{3}{4}|Y_i| \\ &\geq (|X_i| + |Z_i| + |Y_i|)/4 \\ &\geq |Y_{i+1}| \end{aligned}$$

The first statement follows by algebra from equation 11. The third line follows from the induction assumption that $|Y_i| \leq |X_i|$. The last line follows by equation 10.

We now argue that $\max_t(\tilde{U}^4_{FCFS}(t)/\tilde{U}^1_{\text{OPT}}(t)) \leq 4$. Let t be an arbitrary time in some interval I_i.

$$\begin{aligned} \tilde{U}^1_{\text{OPT}}(t) &\geq |X_i| + |Z_i(t)| - (t - z_i) \\ &\geq |X_i| + |Z_i(t)| - (|X_i| + |Y_i| + |Z_i(t)|)/4 \\ &= \frac{3}{4}|X_i| - \frac{1}{4}|Y_i| + |Z_i(t)| - \frac{1}{4}|Z_i(t)| \\ &\geq \frac{3}{4}|X_i| - \frac{1}{4}|Y_i| + \frac{3}{4}|Z_i(t)| \\ &\geq \frac{3}{4}|X_i| - \frac{1}{4}|Y_i| + \frac{1}{4}|Z_i(t)| \\ &\geq \frac{1}{4}|X_i| + \frac{1}{4}|Y_i| + \frac{1}{4}|Z_i(t)| \\ &\geq \tilde{U}^4_{FCFS}(t)/4 \end{aligned}$$

The first line follows since the adversary has a unit speed server. Then second line follows since the duration of the interval $[z_i, t]$ is at most $(|X_i|+|Y_i|+|Z_i(t)|)/4$. The fifth line follows since $|Y_i| \leq |X_i|$. And the last line follows because $\tilde{U}^4_{FCFS}(t) \leq |X_i| + |Z_i(t)| + |Y_i|$. ∎

We now show that the above lemma is essentially tight by showing that FCFS is not a 3-speed $O(1)$-approximation algorithm with respect to minimizing the number of pages with unsatisfied requests.

Lemma 12 $\max_t(\tilde{U}^3_{FCFS}(t)/\tilde{U}^1_{\text{OPT}}(t)) = \omega(1)$.

Proof: Let $n = 9k + 9$ for some integer k. The adversary requests each page once at time 0. Without loss of generality, assume that the initial order of pages in FCFS's queue Q from front to back is

$P_1, \ldots, P_n$. The adversary does not broadcast a page at time 1. At each time $t \geq 1$, the adversary requests those pages P_j such that, P_j was broadcasted by FCFS at time t, and the adversary has unsatisfied requests for P_j at time t. At each time $t \geq 2$, the adversary will broadcast its unsatisfied page P_j that was most recently broadcast by FCFS strictly before time t, breaking ties in favor of higher numbered pages (we will see that P_j must have been broadcast by FCFS at time $t-1$).

Let z_i be the first time that FCFS has broadcast some page exactly i times. So $z_0 = 0$. We say that a page P_j is type X at time t, if there is an unsatisfied request for P_j for both FCFS and the adversary at time t. We say that a page P_j is type Y at time t, if there is an unsatisfied request for P_j for FCFS, but not for the adversary, at time t. Note that by the adversarial strategy, every page must either by type X or type Y. Hence, the state of both FCFS and the adversary can be specified by specifying the sequence of X/Y types in Q.

By simulating FCFS and the adversary's strategy one can verifying the following claims. If you examine the types of the pages in Q at time z_1, from front to back, the sequence is $(XXY)^{3k+2}XXX$. If you examine the types of the pages in Q at time z_2, from front to back, the sequence is $(XY)^{3k+2}XX = (XYXYXY)^k XYXYXX$. If you examine the types of the pages in Q at time z_3, from front to back, the sequence is $(XYY)^k XYXX$. If you examine the types of the pages in Q at time $z_4 - 1$, from front to back, the sequence is XY^kXX (Notice that the X on the front of the queue has not been broadcast a fourth time, while all other pages have). Hence, at time $z_4 - 1$ the adversary has only 3 pages with unsatisfied requests, while FCFS has $\Omega(n)$ pages with unsatisfied requests. ■

We now turn our attention to MRF and LWF.

Lemma 13 For all speeds s, $\max_t(\tilde{U}^s_{MRF}(t)/\tilde{U}^1_{\text{OPT}}(t)) = \omega(1)$, and $\max_t(\tilde{U}^s_{LWF}(t)/\tilde{U}^1_{\text{OPT}}(t)) = \omega(1)$.

Proof: Consider the following input. At time 0, page P_1 is not requested, and every other page is requested once. From time 1 until time n, there are n^2 requests to page P_1 at each time, and no requests for the other pages. At each time t, $2 \leq t \leq n$, both MRF and LWF will broadcast page P_1. So at time n, both MRF and LWF will have $n-1$ pages with unsatisfied requests. Note, however, that by time n it is possible for the adversary to satisfy all requests except for those to page P_1. ■

3.4 Maximizing the Number of Satisfied Requests

We now show that MRF is a 2-competitive with respect to maximizing the number of satisfied requests. Recall that $S_A(t)$ is the number of requests satisfied by algorithm A by time t using a unit speed server on some generic input.

Lemma 14 For all inputs, $\max_t(S_{\text{OPT}}(t)/S_{MRF}(t)) \leq 2$.

Proof: We use an amortization argument. Assume that MRF earns two points for each request it satisfies. The theorem then follows if we show that at all times t it is the case $S_{\text{OPT}}(t)$ is at most the points earned by MRF up until time t. Whenever MRF satisfies a request $R_i(t)$ at time $C_i(t)$, it distributes the $2R_i(t)$ points that it earns at time $C_i(t)$ in the following way:

1. MRF immediately credits itself with $R_i(t)$ points at time $C_i(t)$.
2. If OPT satisfies requests $R_i(t)$ at some time $s > C_i(t)$, then MRF credits itself with the other $R_i(t)$ points at time s.

We now prove by induction on t that $S_{\mathrm{OPT}}(t)$ is at most the points that MRF has credited to itself up until time t. The base case $t = 0$ is trivial, since no requests can be satisfied by time 0. Now suppose that the claim holds for time t, and consider time $t+1$. At time $t+1$, suppose that the adversary satisfies x requests for some page P_j. Let y be the number of these x requests satisfied by MRF prior to time $t+1$. The algorithm MRF credits itself with y requests at time $t+1$ by rule 2. The algorithm MRF could satisfy $x-y$ requests by broadcasting page P_j at time $t+1$. Therefore, by the definition of MRF, MRF satisfies at least $x-y$ requests at time $t+1$. Hence, MRF credits itself with $x-y$ points at time $t+1$ by rule 1. Hence, MRF credits itself with x points at time $t+1$. Therefore, MRF has credited itself with at least $S_{\mathrm{OPT}}(t+1)$ by time $t+1$, and we are done. ■

We now show that LWF and FCFS not $O(1)$-competitive with respect to maximizing the number of satisfied requests.

Lemma 15 $\max_t(S_{\mathrm{OPT}}(t)/S_{LWF}(t)) = \omega(1)$.

Proof: Let $n = 2m$. The adversary requests each page once at time 0. Then the adversary requests no pages until time m. Assume that LWF broadcasted pages $P_1, \ldots, P_m$ by time m. At time $m+t$, $0 \leq t \leq m-1$, the adversary makes $m/(m-t)$ requests for page P_{t+1}.

Observe that LWF will broadcast pages P_{m+1} to P_{2m} from time $m+1$ to time $2m-1$. To see this, consider time $m+t$, $1 \leq t \leq m-1$. The total waiting time for pages $P_{m+t} \ldots P_{2m}$ is $m+t$. The total waiting time for page P_i, $i \leq t$, is $(t-i+1)\frac{m}{m-i}$, which is less than m. At time $2m-1$, LWF has satisfied only $2m-1$ requests. While the adversary could have satisfied $m(1/m+1/m-1+\ldots+1/3) = \Theta(m \log m)$ requests by time $2m-1$. The adversary could have accomplished this by broadcasting pages $P_{m+1} \ldots P_{2m}$ during the first m time units, and then broadcasting page P_t, $1 \leq t \leq m$ at time $m+t+1$. ■

Lemma 16 $\max_t(S_{\mathrm{OPT}}(t)/S_{FCFS}(t)) = \omega(1)$.

Proof: Every page except P_1 is requested at time 0. At time 1, page P_1 is requested n^2 times. At time $n-1$, FCFS has satisfied only the $n-1$ requests pages $P_2, \ldots, P_n$, while the adversary could have satisfied n^2 requests by broadcasting P_1 sometime before time $n-1$. ■

4 Conclusion

There are two obvious interesting directions for further investigation. The first direction is to determine if the problem of minimizing the average response time is NP-complete, and if so, to determine how well the objective function can be approximated by a polynomial time algorithm. For example, is there a polynomial time $O(1)$-approximation algorithm, or better a polynomial time approximation scheme? The second direction of investigation is to determine whether LWF is an $O(1)$-speed $O(1)$-approximation algorithm, and if it is not, determine whether any such algorithm exists.

References

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

[2] AirMedia website, http://www.airmedia.com.

[3] Y. Bartal, and S. Muthukrishnan, "Minimizing Maximum Response Time in Scheduling Broadcasts", ACM/SIAM Symposium on Discrete Algorithms, 558-559, 2000.

[4] E. Coffman, and P. Denning, *Operating Systems Theory*, Prentice-Hall, 1973.

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

[6] Hybrid website, http://www.hybrid.com.

[7] Intel intercast website, http://www.intercast.com.

[8] B. Kalyanasundaram, and K. Pruhs, "Speed is as powerful as clairvoyance", *IEEE Symposium on Foundations of Computation*, 214 – 221, 1995.

[9] E. Lawler, J.K. Lenstra, A. Rinnooy Kan, and D. Shmoys, "Sequencing and Scheduling: Algorithms and Complexity", in S. Graves, A. Rinnooy Kan, and P. Zipkin (eds.), *Logistics of Production and Inventory*, Handbooks in OR & MS 4, Elsevier Science, Chapter 9, 445–522, 1993.

[10] S. Leonardi, and D. Raz, "Approximating total flow time on parallel machines", *ACM Symposium on Theory of Computing*, 110 – 119, 1997.

[11] C. Phillips, C. Stein, E. Torng, and J. Wein "Optimal time-critical scheduling via resource augmentation", *ACM Symposium on Theory of Computing*, 140 – 149, 1997.