Improved approximation algorithms for broadcast scheduling

Citation for published version (APA):

Bansal, N., Coppersmith, D., & Sviridenko, M. (2008). Improved approximation algorithms for broadcast scheduling. *SIAM Journal on Computing*, 38(3), 1157-1174. <https://doi.org/10.1137/060674417>

DOI:

[10.1137/060674417](https://doi.org/10.1137/060674417)

Document status and date:

Published: 01/01/2008

Document Version:

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy

If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

IMPROVED APPROXIMATION ALGORITHMS FOR BROADCAST SCHEDULING*

NIKHIL BANSAL[†], DON COPPERSMITH[‡], AND MAXIM SVIRIDENKO[†]

Abstract. We consider scheduling policies in a client-server system where the server delivers data by broadcasting it to the users. In the simplest model of the problem, there is a single server that holds n pages of unit size. Multiple requests for these pages arrive over time. At each time slot the server broadcasts exactly one page which satisfies all of the outstanding requests for this page at that time. We consider the problem of minimizing the average response time of requests, where the response time of the request is the duration since the request is placed until the time it is satisfied. For the offline version of this problem we give an algorithm with an approximation ratio of $O(\log^2(n)/\log \log(n))$. More generally, for any $\epsilon > 0$, the algorithm achieves an average response time of $(2 + \epsilon) \cdot \text{OPT} + O(\log n \cdot \log_{(1+\epsilon)} n)$, which is useful when the optimum value is large. This substantially improves the previously best known approximation factor of $O(\sqrt{n})$ for the problem [N. Bansal, M. Charikar, S. Khanna, and J. Naor, *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, Vancouver, British Columbia, ACM, New York, SIAM, Philadelphia, 2005, pp. 215–221]. Our result is based on iteratively relaxing and rounding an auxiliary linear program derived from a natural linear programming relaxation of the problem.

Key words. LP rounding, approximation algorithm, broadcast scheduling

AMS subject classifications. 68W25, 68M20, 68Q25, 68W40, 90B35, 90C59

DOI. 10.1137/060674417

1. Introduction. In a broadcast data dissemination system, we have a collection of data items at a broadcast server (typically a satellite). Users submit requests for these data items at various times and the server continuously transmits the data items. Whenever the server broadcasts a data item, it simultaneously satisfies all users waiting for that item. Broadcast systems have received a lot of attention recently. These systems exploit the fact that most requests are for a small common set of objects and they scale very well with increasing demand and number of users. Data broadcasting is actually being used in many commercial systems such as Intel Intericast system, Hughes DirecPC system [10], and the Airmedia system [1] to increase the bandwidth of the system. In fact broadcast is not unique to computer systems. There are several radio music channels where listeners submit the requests for songs, and then these songs eventually get played.

There has been a lot of research interest in various aspects of broadcast systems. In this paper we consider scheduling algorithms to improve the quality of service perceived by the users of the system. We focus on the average response time, which is one of the most commonly used measures of quality of service, defined as the average time a user waits until his request is satisfied.

Problem formulation. The setting and problem we study in this paper is formalized as follows: There is a collection of pages $P = \{1, \dots, n\}$. Time is slotted and any page can be broadcast in a *single* time slot. At any time t , the broadcast server

*Received by the editors November 9, 2006; accepted for publication (in revised form) March 6, 2008; published electronically July 16, 2008.

<http://www.siam.org/journals/sicomp/38-3/67441.html>

[†]IBM T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598 (nikhil@us.ibm.com, sviri@us.ibm.com).

[‡]IDA Center for Communications Research, Princeton, NJ 08540 (dcopper@idacrr.org). This work was done while this author was at IBM T.J. Watson Research Center.

receives $n_p(t)$ requests for page $p \in P$. We say that a request ρ for page p that arrives at time t is satisfied at time $c_p(t)$, if $c_p(t)$ is the first time after t when page p is transmitted by the broadcast server. The *response time* of the request ρ is defined to be the time that elapses from its arrival till the time it is satisfied, i.e., $c_p(t) - t$. We assume that request ρ arrives in the end of the time slot t and, therefore, cannot be satisfied in the time slot in which it arrived; i.e., the response time for any request is at least 1. Let T denote the last time when any request arrives.

We consider the average response time minimization problem, where we want to find a broadcast schedule that minimizes the average response time, defined to be $(\sum_{p \in P} \sum_{t=1}^T n_p(t)(c_p(t) - t)) / (\sum_{p \in P} \sum_{t=1}^T n_p(t))$. In this paper we study the offline problem, where the request sequence is known in advance to the scheduling algorithm.

The problem can also be viewed as the response time minimization version of the minimum latency problem (aka traveling repairman problem) on a uniform metric space. In the classical minimum latency problem [5, 8, 16], we wish to find a tour on n points in a metric space which minimizes the average time a vertex is visited. This can be viewed as a request arriving at each vertex at time 0, and the goal is to satisfy the average response time of the requests. One can consider a stronger variant of the minimum latency problem when requests at vertices arrive over time, and, moreover, several requests might arrive at a vertex at different times. By associating each vertex with a page, and viewing every visit to a vertex as a transmission of the corresponding page, it can be seen that the problem considered in this paper is identical to the response time version of the latency problem for the uniform metric space.

Previous work. The average response time problem was shown to be NP hard by Erlebach and Hall [13]. Most of the previous algorithmic work has focused on resource augmentation where the server is given extra speed compared to the optimal algorithm. These results compare k -speed approximation algorithm against the performance of an optimal 1-speed algorithm, where a k -speed algorithm is one that allows a server to broadcast k pages in each time slot. Kalyanasundaram, Pruhs, and Velauthapillai [17] gave the first $\frac{1}{\alpha}$ -speed, $\frac{1}{1-2\alpha}$ -approximation algorithm for any fixed α , for $0 \leq \alpha \leq 1/3$. This guarantee was improved in a sequence of papers [14, 13, 15]. Gandhi et al. [14] gave a $\frac{1}{\alpha}$ -speed, $\frac{1}{1-\alpha}$ -approximation algorithm for any $\alpha \in (0, 1/2]$. Erlebach and Hall [13] gave a 6-speed 1-approximation algorithm for the problem, which was improved to a 4-speed, 1-approximation algorithm by [14]. Then, Gandhi et al. [15] gave a 2-speed, 1-approximation. Subsequently, Bansal et al. [2] gave an algorithm that achieves a constant approximation ratio for arbitrarily small extra speed up factor. Their algorithm achieves an additive approximation ratio of $O(1/\epsilon)$ with $(1 + \epsilon)$ -speed.¹

When no extra speed is allowed, the problem seems to be considerably harder. Note that repeatedly transmitting the pages in the cyclic order $1, \dots, n$ is an $O(n)$ approximation, as every request has a response time of at most n in the schedule above, and at least 1 in any schedule. Prior to our work, the only result with a guarantee better than the naive $O(n)$ was due to Bansal et al. [2]. Their algorithm produces a solution with an average response time of $\text{OPT} + O(\sqrt{n})$, where OPT is the optimum average response time. We note that this is an additive $O(\sqrt{n})$ approximation, hence it directly implies an $O(\sqrt{n})$ multiplicative approximation (since OPT is at least 1). Alternately, the result can be viewed as an $O(\max(1, \sqrt{n}/\text{OPT}))$ multiplicative

¹Here $(1 + \epsilon)$ -speed means that the algorithm is allowed to transmit one extra page every $1/\epsilon$ time steps.

approximation. Furthermore, for any $\epsilon > 0$, the same algorithm achieves the $(1 + \epsilon)$ -speed, $O(1/\epsilon)$ -additive approximation guarantee mentioned earlier.

In the online setting, a lower bound of $\Omega(\sqrt{n})$ without speedup and a lower bound of $\Omega(1/\epsilon)$ with a speedup factor of $(1 + \epsilon)$ on the competitive ratio of any randomized online algorithm is known [2]. In [17, Lemma 7], an $\Omega(n)$ lower bound on the competitive ratio of deterministic algorithms is given. Edmonds and Pruhs [11] gave a $(4 + \epsilon)$ -speed, $O(1 + 1/\epsilon)$ -competitive online algorithm. Later, they [12] showed that a natural algorithm, Longest Wait First, is 6-speed, $O(1)$ -competitive. Recently, Robert and Schabanel have extended these results to the setting where jobs can have dependencies [19].

Other measures have also been studied in the broadcast setting. Bartal and Muthukrishnan [4] considered the problem of minimizing the maximum response time of a request and gave an $O(1)$ -competitive algorithm. Charikar and Khuller [7] consider a generalization of the above problem where the goal is to minimize the maximum response time of a certain fraction of the requests. The profit maximization version of the problem has also been studied. Here each request also has a deadline and a profit which is obtained if this request is satisfied before its deadline. This problem was first studied by Bar-Noy et al. [3] (in a more general setting), who gave a $1/2$ -approximation algorithm for it. Later, Gandhi et al. [15] designed a $3/4$ -approximation algorithm for the problem by using the dependent randomized rounding technique. Recently, Chang et al. [6] showed that this problem is NP-hard even in the special case when all profits are equal to 1 (referred to as the throughput maximization problem).

Our results. Our main result is an $O(\log^2 n / \log \log n)$ -approximation algorithm for minimizing average response time in the absence of extra speed. In fact the bound above follows from a more general guarantee of $(2 + \gamma) \cdot \text{OPT} + O(\log_{1+\gamma} n \cdot \log n)$, where $\gamma > 0$ is an arbitrary positive parameter and OPT denotes the value of the optimum solution. The bound $O(\log^2 n / \log \log n)$ follows by choosing $\gamma = \Theta(\log n)$. Setting γ arbitrarily close to 0 implies a guarantee of $(2 + \gamma)\text{OPT} + O(\log^2 n / \gamma)$, which could be more useful in cases when OPT is large. For example, this implies an $O(1)$ approximation when OPT is $\Omega(\log^2 n)$.

Organization. We begin by defining a natural integer programming formulation in section 1 and discuss some properties of its linear programming (LP) relaxation. Section 2 describes the high level idea of our algorithm, and we present the algorithm and its analysis in section 3. The analysis is split into two parts. For simplicity, we first show a guarantee of $3 \cdot \text{OPT} + O(\log^2(T + n))$, where T is the time horizon. Then, in section 3.5 we show how to remove the dependence on T , and show how to refine the guarantee further to obtain an $O(\log^2 n / \log \log n)$ approximation. Finally, in section 4 we give an example of a solution to the LP relaxation where every local rounding procedure (defined below) incurs a large gap. This essentially implies that techniques used prior to our work are unlikely to yield an approximation ratio better than $\Theta(\sqrt{n})$.

Preliminaries. We begin by considering an LP relaxation of a natural time-indexed integer linear program (ILP) for the response time minimization problem. This formulation is also the starting point of all previously known approximation algorithms for this problem [17, 14, 15, 2].²

Consider the following time-indexed integer programming formulation: For each page $p = 1, \dots, n$ and each time t' , there is a variable $y_{pt'}$ which indicates whether

²Strictly speaking, [17] considered a different LP formulation. However, it is equivalent [18] to the one considered in this paper.

page p was transmitted at time t' . In particular, $y_{pt'}$ is 1 if p is broadcast at time t' and 0 otherwise. We have another set of variables $x_{ptt'}$ indexed by a page p and two times t and t' such that $t' > t$. These are used to model the response of a request. In particular, $x_{ptt'} = 1$ if a request for page p that arrived at time t is satisfied at time $t' > t$ and 0 otherwise. Let n_{pt} denote the number of requests for page p that arrive at time t . The following integer program is an exact formulation of the problem:

$$\begin{aligned}
 (1) \quad & \min \sum_p \sum_t \sum_{t'=t+1}^{T+n} (t' - t) \cdot n_{pt} \cdot x_{ptt'} \\
 (2) \quad & \text{subject to } \sum_p y_{pt'} \leq 1 \quad \forall t', \\
 (3) \quad & \sum_{t'=t+1}^{T+n} x_{ptt'} \geq 1 \quad \forall p, t, \\
 (4) \quad & x_{ptt'} \leq y_{pt'} \quad \forall p, t, t' > t, \\
 (5) \quad & x_{ptt'} \in \{0, 1\} \quad \forall p, t, t', \\
 (6) \quad & y_{pt'} \in \{0, 1\} \quad \forall p, t'.
 \end{aligned}$$

Here T denotes the last time when any request arrives. Observe that it suffices to define variables only until time $t = T + n$, as all of the requests can be satisfied by transmitting page p at time $T + p$ for $p = 1, \dots, n$. The set of constraints (2) ensures that at most one page is transmitted in each time slot. The set of constraints (3) ensures that each request must be satisfied and, finally, the set of constraints (4) ensures that a request for page p can be satisfied at time t only if p is transmitted at time t . A request that arrives at time t and is satisfied at time t' contributes $(t' - t)$ to the objective function. Without loss of generality, it can be assumed that in any optimum solution the constraints (2) are satisfied with an equality and that for every p and t , the variable $x_{ptt'}$ is equal to 1 at the earliest time $t' > t$ such that $y_{pt'} = 1$.

As stated the size of the formulation is polynomial in the time horizon T which could be arbitrarily large. However, we can assume that the input size of the problem is at least T/n , since if there is a period of n consecutive time units when no page is requested, then we can split the problem into two disjoint instances.

Consider the linear program obtained by relaxing the integrality constraints on $x_{ptt'}$ and $y_{pt'}$. This fractional relaxation may be viewed as broadcasting pages fractionally at each unit of time such that total fraction of all the pages broadcast in any unit of time is 1. A request for a page p arriving at a time t is considered completely satisfied at time t' if t' is the earliest time such that the total amount of page p broadcast during the interval $(t, t']$ is at least 1.

Let (x^*, y^*) be some optimum solution to the fractional relaxation of the program (1)–(6). It is easily seen that the value of variables (x^*) are completely determined by the values of variables (y^*) . Since each page needs one unit of transmissions to be satisfied, we can assume that for each p and t , the variable $x_{ptt'}^*$ is greedily set equal to $y_{pt'}^*$ starting from $t' = t + 1$ until one unit of page p has been transmitted. Equivalently, for each page p , time t , and any time $\tau > t$,

$$(7) \quad \sum_{t'=\tau}^{\infty} x_{ptt'}^* = \max \left\{ 0, \left(1 - \sum_{t''=t+1}^{\tau-1} y_{pt''}^* \right) \right\}.$$

For each page p and time t , the solution (x^*, y^*) determines a (fractional) response time $r(p, t)$ for the request for p that arrives at time t . In particular, $r(p, t) = \sum_{t' > t} (t' - t) x_{ptt'}^*$. As $(t' - t)$ can be written as $\sum_{\tau=t+1}^{t'} 1$, we can rewrite $r(p, t) = \sum_{t' > t} \sum_{\tau=t+1}^{t'} x_{ptt'}^*$. By interchanging the order of summation, we get that $r(p, t) = \sum_{\tau > t} \sum_{t' \geq \tau} x_{ptt'}^*$. By (7), it follows that for each page p and time t we have

$$(8) \quad r(p, t) = \sum_{\tau > t} \max \left\{ 0, 1 - \sum_{t''=t+1}^{\tau-1} y_{pt''} \right\}.$$

The following is an extremely useful view of response times that we will use repeatedly.

LEMMA 1. *Consider a request for a page p that arrives at time t . For $\alpha \in (0, 1]$, let $t(\alpha, p)$ denote the earliest time after t such that the cumulative amount of page p broadcast by the LP during the interval $[t + 1, t(\alpha, p)]$ is at least α . Then $r(p, t) = \int_0^1 (t(\alpha, p) - t) d\alpha$.*

Equivalently, if we choose α uniformly at random in $(0, 1]$ and transmit page p at time $t(\alpha, p)$, then the expected response time for this request is equal to the LP cost for this request.

Proof. Consider some time $\tau > t$. Since α is chosen uniformly at random in $(0, 1]$, the probability that $t(\alpha, p) \geq \tau$ is exactly equal to the probability that $\alpha > \sum_{t''=t+1}^{\tau-1} y_{pt''}$, which is exactly equal to $\max\{0, 1 - \sum_{t''=t+1}^{\tau-1} y_{pt''}\}$. Now,

$$\begin{aligned} \int_0^1 (t(\alpha, p) - t) d\alpha &= \mathbb{E}_\alpha[t(\alpha, p) - t] \\ &= \sum_{\tau > t} \Pr[t(\alpha, p) \geq \tau] \\ &= \sum_{\tau > t} \max \left\{ 0, 1 - \sum_{t''=t+1}^{\tau-1} y_{pt''} \right\} \end{aligned}$$

which is exactly equal to $r(p, t)$ by (8), and hence the result follows. $\square$

2. Algorithm overview and techniques. The algorithm begins by solving the LP relaxation of (1)–(6). Our high level approach is the same as the one introduced in [2]. We first produce a *tentative* schedule which has a good response time, but can violate the “capacity” constraints (2), by transmitting more than one page during certain time slots. However, this violation will be bounded as explained in condition 2 below. In particular, suppose this tentative schedule satisfies the following properties:

1. The total response time for this schedule is at most $c = O(1)$ times the cost of the LP relaxation.
2. The capacity constraints are satisfied approximately in the following sense. For any interval of time $(t, t']$, the total number of pages broadcast by the tentative schedule during $(t, t']$ is at most $t' - t + b$, for some b independent on the time interval $(t, t']$. We refer to this b as the *backlog* of the schedule.

In this case, the tentative schedule can be transformed into a valid schedule as follows: We transmit pages in the same order as the tentative schedule while ensuring that no page is transmitted at an earlier time than in the tentative schedule. It is not hard to see that the backlog property ensures that no page is transmitted more than b steps later than in the tentative schedule (see Lemma 8 for a formal proof). Thus, this

produces an integral solution with average response time $c \cdot \text{OPT} + b$. The $O(\sqrt{n})$ -approximation algorithm of [2] was based on obtaining a tentative solution with cost equal to the LP cost (i.e., $c = 1$) and backlog $O(\sqrt{n})$.

In this paper we will give a procedure to obtain a tentative schedule with $c = 2 + \gamma$ and $b = O(\log_{1+\gamma}(T + n) \cdot \log(T + n))$, for any arbitrary $\gamma > 0$. By Lemma 11, this will imply the desired approximation guarantee of $(2 + \gamma)\text{OPT} + O(\log_{1+\gamma} n \cdot \log n)$. Our improved approximation is based on two new ideas: First, we relax a locality requirement (explained below) in the rounding procedure. Prior to our work, all algorithms were *local* in the following sense: Given a solution to the LP formulation, they produced a schedule (or a tentative schedule) that ensured that for every interval, if the LP solution transmits more than one unit of page p during this interval, then the rounded solution has at least one transmission of p during this interval. The main reason for enforcing this locality was that the response time for each request can be charged directly to its cost in the LP solution, which makes the analysis relatively simple. Interestingly, in section 4 we show that relaxing this locality requirement seems necessary to obtain approximation ratios better than $\Theta(\sqrt{n})$. In particular, we show that there exist LP solutions such that any rounding procedure that is local must have backlog at least $\Omega(\sqrt{n})$.

In our rounding procedure we relax the locality requirement so that it does not necessarily hold for all time intervals. In particular, for each page p we partition the time horizon, $1, \dots, T + n$, into intervals $B(p, i)$ called blocks, where $B(p, i)$ refers to the i th block for page p . These intervals are small in the sense that the cumulative amount of page p transmitted by the LP during $B(p, i)$ is $O(\log(T + n))$. We require only that the rounding be local within each block $B(p, i)$. While this could lead to some requests (that lie at the interface of two blocks) to pay much more than their LP costs, our technique for constructing $B(p, i)$ ensures that the response times of these requests are not too much more.

The second part of the algorithm is to give a scheme to choose local schedules for each block $B(p, i)$ such that when the tentative schedule is constructed by merging these local schedules for all the pages, it satisfies the following two properties. First, the cost of the tentative schedule is bounded by a constant times the optimum, and second, the backlog of the tentative schedule is bounded by $O(\log^2(T + n))$. To do this, we solve a sequence of linear programs iteratively. We begin by defining a (different) linear program (LP') where the variables correspond to the possible local schedules that can be chosen for each block $B(p, i)$. LP' has the property that there is a feasible fractional solution where the backlog is 0 and the response time is only $O(1)$ times the optimum cost. Observe that a solution to LP' can be viewed as assigning a linear combination of local schedules to each block. Our novel step is to show that the number of constraints in LP' can be reduced in such a way that in any basic solution to this reduced linear program, at least a constant fraction of blocks gets assigned exactly one local schedule, and yet the backlog increases only by $O(\log(T + n))$. Since this linear program has fewer constraints, the objective function can only be better. Since at least a constant fraction of blocks gets assigned exactly one schedule, we can remove these blocks to obtain a smaller problem instance. This allows us to apply this procedure iteratively, where we successively relax the constraints. At each iteration, the problem size decreases geometrically, the backlog increases additively by $O(\log(T + n))$, and the objective function does not worsen. After $O(\log(T + n))$ iterations the procedure terminates and we obtain a tentative schedule with backlog $O(\log^2(T + n))$, and cost equal to $O(1)$ times the optimum cost. We now describe the

details.

3. Minimizing average response time. Our algorithm begins by solving the LP (1)–(6). Fix an optimum solution (x^*, y^*) to this LP, and let $r(p, t)$ denote the response time according to the LP solution for a request for page p at time t , that is $r(p, t) = \sum_{t' > t} (t' - t)x_{ptt'}^*$. Let $c(p, t, t')$ denote the cumulative amount of page p transmitted by the LP solution during the time interval $(t, t']$, that is $c(p, t, t') = \sum_{t''=t+1}^{t'} y_{pt''}^*$. In the rest of this paper we refer to the time interval $(t-1, t]$ for $t \geq 1$ as the *time slot* t . We now define the key concept of p -good time slots which will be used to form the blocks for each page.

3.1. Blocks and p -good time slots.

DEFINITION 2 (p -good time slot). *Let $r(p, t)$ denote the response times as determined by the solution to the LP (1)–(6). For a page p we say a time slot t is p -good if $r(p, t) \leq 2r(p, \tau)$ for all $\tau < t$ such that $c(p, \tau, t) \leq 1$.*

Intuitively, the LP response time for page p at a p -good time t is not much more than that at any other time τ , where τ is sufficiently close to t in the sense defined above. The following lemma shows that a p -good slots can be found in any interval of time that broadcasts a sufficient amount of page p .

LEMMA 3. *Any time interval $(t, t']$ such that $c(p, t, t') > \log(T + n)$ contains a p -good slot.*

Proof. For the sake of contradiction, suppose that none of the slots in the interval $(t, t']$ are p -good. Since t' is not p -good, there exists a $t_1 < t'$ such that $c(p, t_1, t') \leq 1$ and $r(p, t_1) > 2r(p, t')$. Note that t_1 lies in the interval (t, t') and, hence, is not p -good by our assumption. Thus there exists t_2 such that $c(p, t_2, t_1) \leq 1$ and $r(p, t_2) > 2r(p, t_1)$. Repeating the argument for $k = \log(T + n)$ steps, we obtain a sequence of slots $t < t_k < \dots < t_2 < t_1 < t'$ such that $c(p, t_k, t') \leq \log(T + n)$ and $r(p, t_k) > 2^k r(p, t') = (T + n) \cdot r(p, t')$, which is impossible as the response time for any request is bounded between 1 and $(T + n)$. $\square$

Lemma 3 implies that if t is a p -good slot, and such that $c(p, t, T+n) > \log(T + n)$, then there is another p -good slot $t' > t$ such that $c(p, t, t') \leq \log(T + n)$. Thus, for each p we can form a collection

$$G(p) = \{0 = t(p, 0), t(p, 1), t(p, 2), \dots, t(p, b_p) = (T + n)\}$$

of time slots such that the $t(p, i)$ is p -good for $1 \leq i \leq b_p - 1$ and $1 \leq c(p, t(p, i - 1), t(p, i)) \leq 1 + \log(T + n)$ for all $1 \leq i \leq b_p - 1$. The last interval must be such that $1 \leq c(p, t(p, b_p - 1), t(p, b_p)) \leq 2 + \log(T + n)$, and such a collection of slots can be formed by a simple iterative greedy strategy.

Indeed, let $t(p, s)$ be the current p -good point. Let $t(p, s, 1)$ be the next p -good point. By Lemma 3 $c(p, t(p, s), t(p, s, 1)) \leq \log(T + n)$. If $c(p, t(p, s), t(p, s, 1)) \geq 1$, then $t(p, s + 1) = t(p, s, 1)$. Otherwise, let $t(p, s, 2)$ be the next p -good point. By Lemma 3 $c(p, t(p, s, 1), t(p, s, 2)) \leq \log(T + n)$ and therefore $c(p, t(p, s), t(p, s, 2)) \leq 1 + \log(T + n)$. If $c(p, t(p, s), t(p, s, 2)) \geq 1$, then $t(p, s + 1) = t(p, s, 2)$. Otherwise, we define $t(p, s, 3)$ and continue until we either define $t(p, s + 1)$ or hit the end of the planning horizon. If the last interval produced by this greedy strategy has length smaller than 1, then we merge this interval and the second-last interval (this is why we have slack for the size of the last interval).

We call the time intervals $(0 = t(p, 0), t(p, 1)]$, $(t(p, 1), t(p, 2)]$, $\dots$, $(t(p, b_p - 1), t(p, b_p)]$, blocks for page p . Note that there are b_p blocks for page p . We will use $B(p, i) = (t(p, i - 1), t(p, i)]$ to denote the i th block for page p . Let $\mathcal{B}_p$ denote the set

of all blocks for page p , and let $\mathcal{B} = \cup_p \mathcal{B}_p$ denote the set of all blocks. For a block $B(p, i)$, we define its *tail* to be the time slots t such that $c(p, t, t(p, i)) < 1$. That is, the cumulative amount of page p transmitted after time t until the end of the block in which t lies is less than 1. Obviously, there is a tail for each block $B(p, i)$ since $c(p, t(p, i-1), t(p, i)) \geq 1$. Note that if a request for page p arrives during the tail of a block, then it is not satisfied completely within that block by the LP solution.

Let us focus on a particular block, say $B(p, i)$. For $\alpha \in (0, 1]$, $l = 0, 1, 2, \dots$, and block $B(p, i)$, let $t(p, i, l, \alpha)$ denote the earliest time in $B(p, i)$ when an $l + \alpha$ amount of page p has been broadcast since the start of $B(p, i)$; i.e., an $(l + \alpha)$ amount of page p has been broadcast during $(t(p, i-1), t(p, i, l, \alpha)]$ (the time $t(p, i, l, \alpha)$ is defined only if block $B(p, i)$ transmits at least $l + \alpha$ amount of page p). For a given block $B(p, i)$ and $\alpha \in (0, 1]$, let $C(p, i, \alpha)$ denote the set of all time slots $t(p, i, l, \alpha)$ for $l = 0, 1, 2, \dots$

Note that if we transmit page p at slots in $C(p, i, \alpha)$ for some $\alpha \in (0, 1]$, then this is a local schedule for $B(p, i)$ (since we transmit page p whenever the LP transmits one unit of page p during any time interval contained in $B(p, i)$). We say that this local schedule is formed by choosing the *offset* α for block $B(p, i)$. We will only be interested in local schedules for blocks that are obtained by choosing some offset $\alpha(p, i)$ for each block $B(p, i)$ and transmitting page p at all time slots in $C(p, i, \alpha(p, i))$. Suppose we arbitrarily choose some offset $\alpha(p, i)$ for each block $B(p, i)$. We claim that the tentative schedule thus obtained satisfies each request. To see this, observe that each request for page p that arrives in $B(p, i)$ for $1 \leq i \leq b_p - 1$ (i.e., except for the last block for page p) is served within $B(p, i)$ or in $B(p, i+1)$. Finally, as there is at least one unit of page p broadcast in the LP solution after the arrival of the last request for page p (see comment after the definition of LP (1)–(6), all requests for page p that arrive during the last block $B(p, b_p)$ are served within $B(p, b_p)$).

The following lemma shows that if we construct a tentative schedule by choosing an offset independently at random for each block, then the expected total response time is not too high and all of the capacity constraints are satisfied at each time step in expectation.

LEMMA 4. *Suppose for each block $B(p, i) \in \mathcal{B}$, we choose the offset $\alpha(p, i)$ independently and uniformly at random in $[0, 1]$. Then the tentative schedule thus obtained satisfies the following properties:*

1. *The expected number of pages transmitted at any time step t is exactly 1.*
2. *For each request, its expected response time is at most 3 times the cost incurred by it in the LP solution of (1)–(6).*

Proof. Consider a time t and suppose that $t \in B(p, i)$. Since we choose the offset $\alpha(p, i)$ uniformly at random in $(0, 1]$, the probability that page p is transmitted at time t is exactly y_{pt}^* .

For any p , the blocks $B(p, i)$ partition the entire time interval $(0, T + n]$, and hence for each time t there is exactly one block for page p that contains t . Thus, the probability that page p is transmitted at time t in the tentative schedule is exactly y_{pt}^* . Summing up over all of the pages, we have that the expected number of pages transmitted at time t is exactly $\sum_p y_{pt}^*$ which is exactly 1 by constraints (2) in the LP. This proves the first part of the lemma.

Consider a particular block $B(p, i)$, and let ρ be a request for page p that arrives during $B(p, i)$. We say that request ρ is *early* if it does not arrive in the tail of $B(p, i)$, or, equivalently, that $c(p, t_\rho, t(p, i)) \geq 1$. Note that ρ is always served within $B(p, i)$ irrespective of the choice of $\alpha(p, i)$. Since $\alpha(p, i)$ is chosen uniformly at random in $(0, 1]$, by Lemma 1 the expected response time for an early request is exactly $r(p, t)$.

Thus it suffices to focus on the contribution of requests that are not early. Consider a request ρ for page p that arrives at time $t_\rho \in B(p, i)$ such that $c(p, t_\rho, t(p, i)) < 1$. Since $\alpha(p, i)$ is chosen uniformly at random, with probability $c(p, t_\rho, t(p, i))$ this request is served in $B(p, i)$ and with probability $1 - c(p, t_\rho, t(p, i))$ it is served in $B(p, i + 1)$. Since $\alpha(p, i + 1)$ is chosen uniformly at random in $(0, 1]$ and is independent of $\alpha(p, i)$, it follows that conditioned on the event that ρ is served in $B(p, i + 1)$, by Lemma 1 its expected response time is $(t(p, i) - t_\rho) + r(p, t(p, i))$. Thus the overall expected response time of ρ is

$$\begin{aligned}
& \left(\sum_{t''=t_\rho+1}^{t(p,i)} (t'' - t_\rho) \cdot x_{pt_\rho t''}^* \right) + (1 - c(p, t_\rho, t(p, i))) \cdot (t(p, i) - t_\rho + r(p, t(p, i))) \\
&= \left(\sum_{t''=t_\rho+1}^{t(p,i)} (t'' - t_\rho) \cdot x_{pt_\rho t''}^* \right) + \left(\sum_{t''=t(p,i)+1}^{\infty} (t(p, i) - t_\rho) \cdot x_{pt_\rho t''}^* \right) \\
&\quad + (1 - c(p, t_\rho, t(p, i))) \cdot r(p, t(p, i)) \\
&\leq \left(\sum_{t''=t_\rho+1}^{t(p,i)} (t'' - t_\rho) \cdot x_{pt_\rho t''}^* \right) + \left(\sum_{t''=t(p,i)+1}^{\infty} (t'' - t_\rho) \cdot x_{pt_\rho t''}^* \right) \\
&\quad + (1 - c(p, t_\rho, t(p, i))) \cdot r(p, t(p, i)) \\
&\leq \left(\sum_{t''=t_\rho+1}^{\infty} (t'' - t_\rho) \right) \cdot x_{pt_\rho t''}^* + r(p, t(p, i)) \leq r(p, t_\rho) + 2r(p, t_\rho) = 3r(p, t_\rho).
\end{aligned}$$

The first equality follows by observing that

$$\left(\sum_{t''=t(p,i)+1}^{\infty} x_{pt_\rho t''}^* \right) = 1 - c(p, t_\rho, t(p, i)).$$

The final step follows as $t(p, i)$ is a p -good time slot; hence by definition $r(p, t(p, i)) \leq 2r(p, t_\rho)$. $\square$

The following lemma will allow us to consider discrete choices for the offsets $\alpha(p, i)$.

LEMMA 5. *Let $\varepsilon > 0$ be an arbitrary precision parameter. We can assume that $x_{ptt'}$ and y_{pt} are integral multiples of $\delta = \varepsilon/(T + n)^2$. This adds at most ε to the response time of each request.*

Proof. Given an arbitrary LP solution, we simply round down the values of y_{pt} to the closest multiple of δ and modify $x_{ptt'}$ accordingly. We also transmit $\delta \cdot T \leq 1/n$ units of each page p at time $T + n + 1$ to ensure that each request remains completely satisfied.

Observe that each $x_{ptt'}$ is reduced by at most δ . As the response time for a request for page p at time t is $\sum_{t' > t} (t' - t)x_{ptt'}$, the rounding adds at most $T \cdot T \cdot \delta \leq \varepsilon$ to the response time of each request. $\square$

Thus we can assume that all of the offsets $\alpha(p, i)$ are integral multiples of δ for $\delta = \varepsilon/(T + n)^2$. Henceforth, we will use $B(p, i, j)$ to denote the time slots in $C(p, i, \alpha = \delta j)$ (i.e., those obtained by choosing $\alpha(p, i) = \delta j$), where $j = 0, \dots, 1/\delta - 1$. We will call the set of time slots $B(p, i, j)$ a *block-offset*.

3.2. Auxiliary LP. Recall that our goal is to choose exactly one offset for each block in such a way that the total response time of the tentative schedule thus obtained is not too high, and the backlog is small. For this purpose we define the LP (9)–(12).

We have variables z_{pij} that correspond to choosing the offset j for block $B(p, i)$, or equivalently transmitting the page p at times in $B(p, i, j)$ during $B(p, i)$. The parameters $R(B(p, i, j))$ and $w(B(p, i, j), t)$ are defined below.

$$(9) \quad \min \sum_p \sum_i \sum_j R(B(p, i, j)) \cdot z_{pij}$$

$$(10) \quad \text{subject to } \sum_j z_{pij} = 1 \quad \forall p, i,$$

$$(11) \quad \sum_p \sum_i \sum_j w(B(p, i, j), t) \cdot z_{pij} = 1 \quad \forall t,$$

$$(12) \quad z_{pij} \geq 0 \quad \forall p, i, j.$$

Here $w(B(p, i, j), t)$ is an indicator function: $w(B(p, i, j), t)$ is 1 if $t \in B(p, i, j)$ and 0 otherwise. Observe that the constraints (10) require that there is exactly one unit of offsets chosen for each block $B(p, i)$, and the constraints (11) require that for each time slot t the total amount of pages transmitted is exactly 1. Thus, this LP can be viewed as choosing a convex combination of local schedules for each block $B(p, i)$ such that certain global constraints are satisfied.

The objective function of minimizing the total response time is expressed in terms of the variables z_{pij} as follows: For each block $B(p, i)$ we associate a *block-offset response time* $R(B(p, i, j))$ which essentially accounts for the contribution of the block-offset $B(p, i, j)$ to the total response time. Observe that choosing an offset for block $B(p, i)$ can affect the response time of requests for page p that arrive in $B(p, i)$ and possibly the requests that arrive during the tail of $B(p, i-1)$. The block-offset response time $R(B(p, i, j))$ is computed as follows:

1. Let t' denote the earliest time in $B(p, i, j)$. Each request for page p in the tail of the previous block $B(p, i-1)$ contributes $t' - t(p, i-1)$ to $R(B(p, i, j))$. Note that this is the amount of time (restricted to time slots in $B(p, i)$) that any request in $B(p, i-1)$ might possibly have to wait until it is satisfied.
2. For a request ρ for page p that arrives at time t where $t \in B(p, i)$, we do the following. Let t' denote the earliest time such that $t' > t$ and $t' \in B(p, i, j)$. If such a t' does not exist, then we set $t' = t(p, i)$. Then the request ρ contributes exactly $t' - t$ to $R(B(p, i, j))$. Note that this quantity is the contribution to the response time of ρ restricted to the time slots in $B(p, i)$.

This definition of $R(B(p, i, j))$ ensures that for any tentative schedule obtained by choosing one offset for each block, i.e., for any setting of z_{pij} to 0 or 1 subject to the constraints (10), the total response time of this tentative schedule is no more than $\sum_{p,i,j} R(B(p, i, j)) \cdot z_{pij}$.

We next observe that the linear program above has a good fractional solution. In particular, Lemma 4 implies the following about the linear program defined by LP (9)–(12).

LEMMA 6. *There is a feasible solution to the LP (9)–(12) with cost no more than 3 times the cost of the optimum value of the LP (1)–(6).*

Proof. Consider the solution where $z_{pij} = \delta$ for all p, i and $0 \leq j \leq 1/\delta - 1$. This corresponds to choosing an offset uniformly at random for each block $B(p, i)$. Lemma 4 (part 1) implies that the constraints (11) are satisfied.

We now show that the cost of this solution is no more than 3 times the cost of LP (1)–(6). Consider an early request in $B(p, i)$. Since this request is completely served within $B(p, i)$ by Lemma 1, the contribution of this request to the objective function is exactly its response time. For a request for page p that arrives at time t in the tail of $B(p, i - 1)$, its contribution to the objective function corresponding to $\sum_j z_{p,i-1,j} \cdot R(B(p, i - 1, j))$ is exactly $\sum_{t''=t+1}^{t(p,i)} (t'' - t) x_{ptt''}^* + (1 - c(p, t, t(p, i))) \cdot (t(p, i) - t)$ which is at most $r(p, t)$. Similarly, the contribution to $\sum_j z_{p,i,j} \cdot R(B(p, i, j))$ is exactly $r(p, t(p, i - 1))$ (by Lemma 1), which is at most $2r(p, t)$ by the definition of a p -good point and as t lies in tail of $B(p, i - 1)$. $\square$

Observe that a solution to the LP (9)–(12) satisfies the capacity constraints (11) exactly. Hence if this LP had an integral optimum solution, then the backlog of the tentative schedule implied by this solution would be 0; we would then have an exact schedule with cost at most 3 times the optimum. However, this LP in general could have a fractional optimum solutions. The rest of the algorithm will deal with obtaining an integral solution to the above LP by successively relaxing the capacity constraints (11) but still ensuring that the quality of the solution remains reasonably good.

3.3. The algorithm. The idea for our algorithm is the following: Suppose we relax the capacity constraints (11) in the auxiliary LP such that we require them only to hold for time intervals of integer size b' rather than for each time unit. That is, we require only that intervals of size b' contain exactly b' pages, but we do not care how these pages are transmitted during an interval. An important observation is that if this relaxed LP had an integral optimum solution, it would give a tentative schedule with backlog $2b'$ and cost at most 3 times the optimum (and we would obtain an approximation guarantee of $3 \cdot \text{OPT} + 2b'$). Since this is too strong to hope for, we will show something weaker. We will show that for $b' = O(\log(T + n))$, there is an optimum solution to this LP where more than half of the blocks $B(p, i)$ have some z_{pij} set integrally to 1. Armed with this result, we will define a smaller problem instance by removing the blocks $B(p, i)$ that have some $z_{pij} = 1$. We then apply this procedure to this smaller problem by redefining the intervals and the LP suitably and relaxing the capacity constraints of the type (11). We repeat this for $O(\log(T + n))$ steps, until we obtain a problem where only a constant number of blocks remain which can be easily solved by trying all possibilities. At a high level, we add $O(\log(T + n))$ to the backlog at each iteration of this process, which will imply that in the end we obtain a tentative schedule with cost at most 3 times the optimum and backlog of $O(\log^2(T + n))$. We now make these arguments precise.

Before we can describe the algorithm to compute the tentative schedule formally, we need some notation. Let $I = (t_1, t_2]$ be a collection of time slots $t_1 + 1, \dots, t_2$. We refer to I as an interval. The size of I denoted by $\text{Size}(I)$ is defined as $t_2 - t_1$. The weight of an interval with respect to $B(p, i, j)$, which we denote by $w(B(p, i, j), I)$, is the cardinality of the set $B(p, i, j) \cap I$. That is, $w(B(p, i, j), I)$ is the number of time slots in the interval I that belong to $B(p, i, j)$.

Our algorithm will solve a sequence of LP's. At step k , some variables z_{pij} that were fractional (not 0 or 1) at the end of step $k - 1$ get assigned to 1. A partial solution is an assignment where some z_{pij} are set to 1. For a partial solution obtained at the end of step k , and an interval I , let $\text{Used}(I, k)$ denote the number of time slots in this interval used up by z_{pij} that are assigned integrally to 1, i.e., $\text{Used}(I, k) = \sum_{p,i,j} w(B(p, i, j), I)$ such that $z_{pij} = 1$. We will use $\text{Free}(I, k)$ to denote $\text{Size}(I) - \text{Used}(I, k)$.

We now describe the algorithm to compute the tentative schedule.

1. Initialize: We divide the time horizon from $1, \dots, T + n$ into consecutive intervals of size $5 \log(T + n)$. We call this collection of intervals $\mathcal{I}_0$. For all $I \in \mathcal{I}_0$, we define $\text{Used}(I, 0) = 0$ and $\text{Free}(I, 0) = \text{Size}(I) - \text{Used}(I, 0) = \text{Size}(I)$. Let $\tilde{\mathcal{B}}_0$ be the set of all blocks $B(p, i)$, and let $\mathcal{S}_0 = \emptyset$.
2. Repeat the following for $k = 1, \dots$,
 - Consider the following linear program defined iteratively based on $\tilde{\mathcal{B}}_{k-1}$, $\mathcal{I}_{k-1}$, and $\text{Free}(I, k-1)$:

(13)

$$\min \sum_{B(p,i) \in \tilde{\mathcal{B}}_{k-1}} \sum_{j=0}^{1/\delta-1} R(B(p,i,j)) \cdot z_{pij}$$

$$(14) \quad \text{subject to} \quad \sum_{j=0}^{1/\delta-1} z_{pij} = 1 \quad \forall p, i : B(p,i) \in \tilde{\mathcal{B}}_{k-1},$$

$$(15) \quad \sum_{p \in P} \sum_{i=1}^{b_p} \sum_{j=0}^{1/\delta-1} w(B(p,i,j), I) \cdot z_{pij} = \text{Free}(I, k-1) \quad \forall I \in \mathcal{I}_{k-1},$$

$$(16) \quad z_{pij} \geq 0 \quad \forall p, i, j.$$

Note that for $k = 1$ this LP is similar to the LP defined by (9)–(12) except that the constraints in (11) are relaxed to hold only for intervals of size $5 \log(T + n)$ rather than for each time unit.

- Solve this LP and consider some basic solution. Let $\mathcal{P}$ denote the set of blocks $B(p, i)$ such that $z_{pij} = 1$ for some j . Let $\mathcal{S}$ denote the set of block-offset pairs $B(p, i, j)$ such that $z_{pij} = 1$.
- Set $\tilde{\mathcal{B}}_k = \tilde{\mathcal{B}}_{k-1} \setminus \mathcal{P}$. These are precisely the blocks $B(p, i)$ for which z_{pij} is not equal to 1 for any j at the end of step k . Set $\mathcal{S}_k = \mathcal{S}_{k-1} \cup \mathcal{S}$. These are precisely the variables z_{pij} that are integrally set to 1 thus far by the end of step k . For each interval $I \in \mathcal{I}_{k-1}$, recompute

$$\text{Used}(I, k) = \text{Used}(I, k-1) + \sum_{p,i,j: B(p,i,j) \in \mathcal{S}} w(B(p,i,j), I).$$

Note that $\sum_{p,i,j: B(p,i,j) \in \mathcal{S}} w(B(p,i,j), I)$ is exactly the number of pages that are assigned to be transmitted during interval I in step k . Set $\text{Free}(I, k) = \text{Size}(I) - \text{Used}(I, k)$. Essentially, $\text{Free}(I, k)$ denotes the number of free time slots in interval I at the end of step k .

- Finally, we compute the set of intervals $\mathcal{I}_k$ by merging the intervals in $\mathcal{I}_{k-1}$ as follows: Initially $\mathcal{I}_k = \emptyset$. Starting from the leftmost interval in $\mathcal{I}_{k-1}$, merge intervals $I_1, I_2, \dots, I_l \in \mathcal{I}_{k-1}$ greedily to form I until $\text{Free}(I_1, k) + \text{Free}(I_2, k) + \dots + \text{Free}(I_l, k)$ first exceeds $5 \log(T + n)$. We set $\text{Free}(I, k) = \text{Free}(I_1, k) + \text{Free}(I_2, k) + \dots + \text{Free}(I_l, k)$ and $\text{Used}(I, k) = \text{Used}(I_1, k) + \text{Used}(I_2, k) + \dots + \text{Used}(I_l, k)$. By construction, we have that $5 \log(T + n) \leq \text{Free}(I, k) \leq 10 \log(T + n)$. Add I to $\mathcal{I}_k$ and remove $I_1, \dots, I_l$ from $\mathcal{I}_{k-1}$ and repeat the process until the total free space in the intervals in $\mathcal{I}_{k-1}$ is less than $5 \log(T + n)$; hence, we cannot form new intervals. In this case we just merge all of the remaining intervals in $\mathcal{I}_{k-1}$ into one interval and add this final interval to $\mathcal{I}_k$.

- If $|\mathcal{I}_k| = 1$, then the algorithm makes one more iteration and then stops. On this last iteration there is just one constraint of type (15) in the relaxed auxiliary LP. The optimal solution is integral and very easy to define. We choose the best offset for every remaining block; i.e., we define $z_{pij} = 1$ if $R(B(p, i, j)) = \min_s R(B(p, i, s))$ for block $B(p, i)$.

3.4. Analysis.

LEMMA 7. *At each iteration of step 2 in the previous algorithm, the number of blocks $B(p, i)$ that do not have any z_{pij} set to 1 decreases by a constant factor. In particular*

$$|\tilde{\mathcal{B}}_k| \leq 0.6 \cdot |\tilde{\mathcal{B}}_{k-1}| + 1.$$

Proof. The total number of nontrivial constraints (of types (14) and (15)) in the LP at step k is $|\mathcal{I}_{k-1}| + |\tilde{\mathcal{B}}_{k-1}|$. Consider a basic optimal solution of the LP at stage k . Let f_k be the number of nonzero variables that are set fractionally (strictly between 0 and 1), and let g_k denote the number of variables set to 1. Then, since we have a basic solution, we have that $f_k + g_k \leq |\mathcal{I}_{k-1}| + |\tilde{\mathcal{B}}_{k-1}|$. Now, consider the constraints of type (14); if in some block $B(p, i)$ there is no z_{pij} that is set to 1, then there must be at least 2 variables z_{pij} set fractionally, which implies that $f_k/2 + g_k \geq |\tilde{\mathcal{B}}_{k-1}|$. Combining these two facts implies that $g_k \geq |\tilde{\mathcal{B}}_{k-1}| - |\mathcal{I}_{k-1}|$. By definition, as $|\tilde{\mathcal{B}}_k| = |\tilde{\mathcal{B}}_{k-1}| - g_k$, this implies that $|\tilde{\mathcal{B}}_k| \leq |\mathcal{I}_{k-1}|$.

We now upper bound $|\mathcal{I}_{k-1}|$. Let Free_{k-1} denote the total free space at the end of iteration $k-1$, that is, $\sum_{I \in \mathcal{I}_{k-1}} \text{Free}(I, k-1)$. Since each interval, except probably the last one, has at least $5 \log(T+n)$ free spaces, we have that $|\mathcal{I}_{k-1}| \leq \lceil \text{Free}_{k-1} / (5 \log(T+n)) \rceil$. As for any block-offset $B(p, i, j)$ and interval I , the number of time slots $w(B(p, i, j), I)$ is at most $\log(T+n) + 2$, it follows from the constraints (14) and (15) that $\text{Free}_{k-1} \leq (\log(T+n) + 2)|\tilde{\mathcal{B}}_{k-1}| \leq 3 \log(T+n)|\tilde{\mathcal{B}}_{k-1}|$. This implies that

$$|\mathcal{I}_{k-1}| \leq \lceil 0.6 \cdot |\tilde{\mathcal{B}}_{k-1}| \rceil \leq 0.6 \cdot |\tilde{\mathcal{B}}_{k-1}| + 1. \quad \square$$

As $|\tilde{\mathcal{B}}_0| \leq T+n$, by Lemma 7 we have that the algorithm stops after $\log(T+n) + \Theta(1)$ iterations. When our algorithm ends we obtain an assignment of zero-one values to variables z_{pij} . Since in every step of our algorithm we relaxed the LP from the previous step, the cost of this final integral solution is upper bounded by the optimal value of LP (9)–(12), which is at most 3 times the optimal value of LP (1)–(6) by Lemma 6. This solution also provides us with an integral tentative schedule since it gives us an assignment of pages to the time slots.

To actually obtain a proper schedule from this tentative schedule, we look at the pages transmitted in the tentative schedule at time 0 and greedily assign it to the next free slot after time t . Formally, we can view the process of constructing the feasible schedule from the tentative schedule as follows: There is a queue Q . Whenever a page p is tentatively scheduled at time t , we add p to the tail of Q at time t . At every time step, if Q is nonempty, we broadcast the page at the head of Q .

To complete the proof, we show that no page is delayed more than $O(\log^2(T+n))$ than its position in the tentative schedule. Thus it suffices to show that the queue length $Q(t)$ at time t in the above description is always bounded by $O(\log^2(T+n))$ at all times t .

LEMMA 8. *Let $\text{Used}(t_1, t_2)$ denote the pages transmitted during $(t_1, t_2]$ in the tentative schedule. The maximum queue length at any time is bounded by $\max_{t_1 < t_2} (\text{Used}(t_1, t_2) - (t_2 - t_1))$.*

Proof. Let t_2 be the time when the backlog in the queue is maximum, and let b denote this backlog. Consider the last time t_1 was before t_2 the queue was empty. Since t_1 was the last time when the queue was empty, it must be the case that exactly $t_2 - t_1$ pages were transmitted during the interval $(t_1, t_2]$, and hence b is exactly $\text{Used}(t_1, t_2) + t_2 - t_1$. This implies the desired result. $\square$

LEMMA 9. *For every t_1, t_2 , $\text{Used}(t_1, t_2) - (t_2 - t_1) \leq 20 \log^2(T + n) + O(\log(T + n))$.*

Proof. Consider the time interval $(t_1, t_2]$. If $\Lambda \leq \log(T + n) + \Theta(1)$ is the number of iterations of our algorithm, then there are at most 2Λ intervals $[a, b]$ generated by the algorithm which strictly overlap with t_1 , i.e., either $a < t_1 < b$ or strictly overlap t_2 , i.e., $a < t_2 < b$. The total number of pages assigned to these intervals by the tentative schedule is at most $2\Lambda \cdot 10 \log n \leq 20 \log^2(T + n) + O(\log(T + n))$.

All other intervals generated by our algorithm do not strictly overlap with t_1 and t_2 . They are either completely inside $(t_1, t_2]$ or else lie completely outside $(t_1, t_2]$. We claim that by the constraints (15), the total number of pages assigned to the intervals completely contained in $(t_1, t_2]$ is upper bounded by $t_2 - t_1$. This follows from the fact that on each iteration we are allowed to use only time slots which were not occupied by the integral assignments from previous iterations and the total number of pages transmitted in every interval on each iteration is exactly the length of this interval minus the amount of free space which could be used on the next iteration. The lemma thus follows. $\square$

Thus we have the following theorem.

THEOREM 10. *The above algorithm produces a broadcast schedule with average response time at most $3 \cdot \text{OPT} + O(\log^2(T + n))$, where OPT denotes the average response time of the optimum schedule.*

3.5. Improving the approximation guarantee further. We first remove the dependence on T in the approximation ratio. We do this by showing that at the expense of a small loss (a factor of $(1 + o(1))$) in the approximation ratio, the time horizon T can be assumed to be polynomially bounded in n .

LEMMA 11. *We are given an instance I of the broadcast scheduling problem with planning horizon of length T and n pages. Then we can define instances $I_1, \dots, I_s$ such that each I_i has a time horizon of length at most $2n^5$, and the solutions to these satisfy the following properties:*

1. *Let LP^* and LP_i denote the optimal value to the LP (1)–(6) for I and I_i , respectively. Then, $\sum_{i=1}^s LP_i \leq (1 + o(1))LP^*$.*
2. *Given any integral solutions $IP_1, \dots, IP_s$ for $I_1, \dots, I_s$, respectively, we can obtain another solution for I with cost $(1 + o(1))(\sum_{i=1}^s IP_i)$.*

It is easily seen that this lemma allows us to assume that $T = \text{poly}(n)$. In particular, we can apply this decomposition to the original instance I , obtain an $O(\log n)$ approximation to each of these instances, and then use the second property to obtain an integral solution to I .

Proof of Lemma 11. We describe a randomized decomposition procedure which satisfies the properties in expectation. Once we describe this procedure it will be immediately clear how this can be derandomized in polynomial time. We first split the original time interval $[1, T]$ into time intervals $T_1, \dots, T_s$ as follows. Choose an integer $k = 1, \dots, n^5$ uniformly at random, and let $T_1 = [1, k + n^5]$, and $T_i = (k + (i - 1)n^5, k + in^5]$ for $i = 2, \dots, s$. Let $V_i \subseteq T_i$ denote the interval consisting of the last n time steps of T_i for $i = 1, \dots, s$. The instance I_i consists of two types of requests in I : those that arrive during $T_i \setminus V_i$ and those that arrive during V_i . We keep the requests

that arrive during $T_i \setminus V_i$ unchanged and move those that arrive during V_i to I_{i+1} by changing their arrival time to the beginning of T_{i+1} (thus I_i contains the requests that arrive during V_{i-1}). This completes the description of the instances. Note that there are only $O(n^5)$ possible decompositions, and hence the procedure can be easily derandomized by trying out all choices of k .

Consider an optimal solution (x, y) to LP (1)–(6) for I . Suppose we choose this solution for each I_i . Then response times for requests that arrive during $T_i \setminus V_i$ remain unchanged and it can only increase for those that arrive during V_i . To handle this, we modify the solution (x, y) as follows. Consider the first n^2 time slots of T_{i+1} , since there are n pages there must be at least some page p_{i+1} such that the amount of p_{i+1} transmitted by the LP during these slots is at least n . We remove $n - 1$ units of page p_{i+1} from these slots and, for every page other than p_{i+1} , arbitrarily assign one unit of this page fractionally to the free space in the first n^2 slots. This can increase the response time of requests that arrive during T_{i+1} by at most n^2 . Moreover, the response time of requests that arrive during V_i is at most $n^2 + n$. Since a request has a probability of at most $n/n^5 = 1/n^4$, over the random choices of k , of arriving in some V_i , the expected increase in the total response time of requests in V_i is at most $(n^2 + n)m/n^4 = o(m)$, where m is the total number of requests in the instance. Thus the average response time increases by at most $o(1)$. Similarly, for requests in the first n^2 slots of T_{i+1} the average response time increases by at most $n^2 m/n^3 = o(m)$ in expectation. This implies the first part of the lemma.

We now prove the second part of the lemma. Assume we are given an integral solution for each I_i . Since all requests in I_i arrive during $T_i \setminus V_i$ and $|V_i| = n$, we can assume that all the requests in I_i are satisfied during T_i . Consider the integral solution for I obtained by concatenating all of the schedules for I_i . When we consider this schedule for the instance I , the only difference is that the response time for requests that arrived during V_i can have an additional response time of n , since these requests were moved at most n steps in the future to obtain the instances I_i . Again, over the random choices of k , the probability that a request lies in some V_i is $1/n^4$ and hence the expected increase in the average response time is at most $n/n^4 = 1/n^3$. This proves the desired result. $\square$

We next show how to obtain a more refined balance between the multiplicative and additive term in Theorem 10.

LEMMA 12. *For any $\gamma > 0$, there is algorithm that achieves an approximation guarantee of $(2 + \gamma)OPT + O(\log_{1+\gamma}(T + n) \cdot \log(T + n))$ for minimizing the average response time. Choosing γ close to 0 implies a guarantee of $(2 + \gamma) \cdot OPT + O(\log^2(T + n)/\gamma)$.*

Proof. For a fixed $\gamma > 0$, we modify the definition (see Definition 2) of a p -good point such that we call a time t to be p -good if $r(p, t) \leq (1 + \gamma)r(p, \tau)$ for all τ, t such that $c(p, \tau, t) < 1$. With this modification, imitating Lemma 3, we can form blocks $B(p, i)$, where the amount of page p transmitted in a block is at most $\log_{1+\gamma}(T + n) + 2$. Moreover, Lemma 4 now gives us that the expected response time of the tentative schedule obtained is at most $(2 + \gamma)$ times the optimum cost. Now repeat the algorithm in section 3.3 with intervals of size $5 \log_{1+\gamma}(T + n)$ (instead of intervals of size $5 \log(T + n)$). Again we have that the number of blocks reduces by more than a factor of half at each iteration while adding $O(\log_{1+\gamma}(T + n))$ to the backlog. Thus, there are $O(\log(T + n))$ iterations of step 2 of the algorithm, and it follows directly that the backlog of the tentative schedule thus constructed is at most $O(\log_{1+\gamma}(T + n) \cdot \log(T + n))$, which implies the desired guarantee. $\square$

By Theorem 10, Lemma 11, and setting $\gamma = \log(T + n)$ in Lemma 12, we have the following theorem.

THEOREM 13. *There is a polynomial time algorithm that produces a broadcast schedule with average response time at most $O((\log^2 n / \log \log n) OPT)$, where OPT denotes the average response time of the optimum schedule.*

4. Bad instance for local rounding procedures. We show that any way of rounding the LP (1)–(6) that gives a tentative schedule that is local is unlikely to achieve an approximation ratio better than $O(\sqrt{n})$. Recall that a local rounding procedure ensures that there is a transmission of page p in every time interval $[t_1, t_2]$, where the LP transmits at least one unit of page p . One consequence of having a local schedule is that if a page p is broadcast at time t , then the next broadcast of page p must be at some time t' such that the cumulative amount of page p transmitted by the LP solution during $(t, t']$ is no more than 1. We give an example of an LP solution for which every tentative schedule that is local has a backlog of $\Omega(\sqrt{n})$ at some time. This will imply that algorithmic techniques based on local tentative schedules are unlikely to yield an approximation guarantee better than $\Theta(\sqrt{n})$.

We construct a half integral LP solution as follows: Let H be the Hadamard matrix of order n , and J be the matrix of order n with all entries equal to 1. Consider the matrix $A = \frac{1}{2}(H + J)$. The matrix A is a $\{0, 1\}$ matrix where each row A_i (except for the one with all 1's) contains exactly $n/2$ 1's. A well-known property (see, for example, [9, p. 17]) of A is that for any vector $x \in \{-1, +1\}^n$, there is a row with a discrepancy of at least $\sqrt{n}/2$. That is, for each vector $x \in \{-1, +1\}^n$, $|A_i x| \geq \sqrt{n}/2$ for some $1 \leq i \leq n$.

We view these A_i as subsets of $\{1, \dots, n\}$. We also assume that n is a multiple of 4. Let n_i denote the number of 1's in A_i . Let $S_i = n_1 + \dots + n_i$, and let $S_0 = 0$. The LP schedule is constructed as follows: The schedule transmits, in any particular order, $1/2$ unit of each page in A_i during the interval $(S_{i-1}, S_{i-1} + n_i/2]$, and again during the interval $(S_{i-1} + n_i/2, S_i]$. For $j \in A_i$, we denote the time when it is transmitted (half unit) during $(S_{i-1}, S_{i-1} + n_i/2]$ its *odd* slot and its transmission during $(S_{i-1} + n_i/2, S_i]$ its *even* slot.

Now, any local tentative schedule requires that if a page is transmitted in its odd slot at some time t , then the next transmission of this page must be no later than next odd slot for this page, and similarly for even slots. We first note that it suffices to consider *strictly* local schedules where each page is transmitted in the tentative schedule only during the odd slots or only during the even slots. Indeed, if some page is transmitted in both an odd slot and the adjacent even slot, then the number of transmissions of the page in the integral solution will be strictly more than those in the LP solution, leading to a backlog of at least 1. Repeating the instance k times (with completely disjoint set of pages) will imply a backlog of k . Thus it suffices to consider strictly local schedules.

A strict local schedule associates a vector $x \in \{-1, +1\}^n$, where the i th entry is -1 if page i is transmitted during odd slots and is 1 otherwise. As the number of pages transmitted by the tentative schedule during $(S_{i-1}, S_i]$ is exactly n_i , it is easy to see that $|A_i x|$ is exactly equal to the backlog at time $S_{i-1} + n_i/2$ or S_i , which implies the desired claim for strict tentative schedules.

5. Concluding remarks. In the offline setting, nothing more than NP-completeness is known for the problem. It would also be very interesting to construct nontrivial integrality gaps for the LP relaxation considered in this paper. Currently, the best

known integrality gap for the LP relaxation is 1.027 [2], and hence it is possible that an $O(1)$ approximation for the problem can be obtained using this LP.

Another very interesting problem would be to give an $O(1 + \epsilon)$ -speed, $O(1)$ -competitive algorithm for the online version of the problem. Currently, no $O(1)$ -competitive algorithm is known for the problem that uses a speed up of less than 4.

Acknowledgments. We thank Moses Charikar, Sanjeev Khanna, Tomasz Nowicki, Tracy Kimbrel, Seffi Naor, Kirk Pruhs, Baruch Schieber, and Grzegorz Swirszcz for helpful discussions.

REFERENCES

- [1] AIRMEDIA WEBSITE, <http://www.airmedia.com>.
- [2] N. BANSAL, M. CHARIKAR, S. KHANNA, AND J. NAOR, *Approximating the average response time in broadcast scheduling*, in Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, Vancouver, British Columbia, ACM, New York, SIAM, Philadelphia, 2005, pp. 215–221.
- [3] A. BAR-NOY, S. GUHA, Y. KATZ, J. NAOR, B. SCHIEBER, AND H. SHACHNAI, *Throughput maximization of real-time scheduling with batching*, in Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, CA, ACM, New York, SIAM, Philadelphia, 2002, pp. 742–751.
- [4] Y. BARTAL AND S. MUTHUKRISHNAN, *Minimizing maximum response time in scheduling broadcasts*, in Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, CA, ACM, New York, SIAM, Philadelphia, 2000, pp. 558–559.
- [5] A. BLUM, P. CHALASANI, D. COPPERSMITH, W. R. PULLEYBLANK, P. RAGHAVAN, AND M. SUDAN, *The minimum latency problem*, in Proceedings of the 26th Annual Symposium on Theory of Computing (STOC), Montreal, Quebec, 1994, pp. 163–171.
- [6] J. CHANG, T. ERLEBACH, R. GAILIS, AND S. KHULLER, *Broadcast scheduling: Algorithms and complexity*, in Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, CA, ACM, New York, SIAM, Philadelphia, 2008, pp. 473–482.
- [7] M. CHARIKAR AND S. KHULLER, *A robust maximum completion time measure for scheduling*, in Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, Miami, FL, ACM, New York, SIAM, Philadelphia, 2006, pp. 324–333.
- [8] K. CHAUDHURI, B. GODFREY, S. RAO, AND K. TALWAR, *Paths, trees, and minimum latency tours*, in Proceedings of the 44th Annual Symposium Foundations of Computer Science (FOCS), Cambridge, MA, 2003, pp. 36–45.
- [9] B. CHAZELLE, *The Discrepancy Method: Randomness and Complexity*, 1st ed., Cambridge University Press, Cambridge, 2000.
- [10] DIREPC WEBSITE, <http://www.direpc.com>.
- [11] J. EDMONDS AND K. PRUHS, *Broadcast scheduling: When fairness is fine*, in Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, CA, ACM, New York, SIAM, Philadelphia, 2002, pp. 421–430.
- [12] J. EDMONDS AND K. PRUHS, *A maiden analysis of longest wait first*, in Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, LA, ACM, New York, SIAM, Philadelphia, 2004, pp. 818–827.
- [13] T. ERLEBACH AND A. HALL, *NP-hardness of broadcast scheduling and inapproximability of single-source unsplittable min-cost flow*, in Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, CA, ACM, New York, SIAM, Philadelphia, 2002, pp. 194–202.
- [14] R. GANDHI, S. KHULLER, Y. KIM, AND Y.-C. WAN, *Approximation algorithms for broadcast scheduling*, in Proceedings of the 9th Annual Conference on Integer Programming and Combinatorial Optimization (IPCO), Cambridge, MA, 2002.
- [15] R. GANDHI, S. KHULLER, S. PARTHASARATHY, AND A. SRINIVASAN, *Dependent rounding in bipartite graphs*, J. ACM, 53 (2006), pp. 324–360.
- [16] M. GOEMANS AND J. KLEINBERG, *An improved approximation ratio for the minimum latency problem*, Math. Programming, 82 (1998), pp. 111–124.
- [17] B. KALYANASUNDARAM, K. PRUHS, AND M. VELAUTHAPILLAI, *Scheduling broadcasts in wireless networks*, in Proceedings of the 8th Annual European Symposium on Algorithms,

- Saarbrücken, Germany, 2000, pp. 290–301.
- [18] S. KHULLER AND Y. KIM, *Equivalence of two linear programming relaxations for broadcast scheduling*, Oper. Res. Lett., 32 (2004), pp. 473–478.
 - [19] J. ROBERT AND N. SCHABANEL, *Pull-based data broadcast with dependencies: Be fair to users, not to items*, in Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, LA, ACM, New York, SIAM, Philadelphia, 2007, pp. 238–247.