Energy-Efficient Broadcast Scheduling for Speed-Controlled Transmission Channels

Patrick Briest* Christian Gunia†

Abstract

We consider the problem of computing energy-efficient broadcast schedules for a speed-controlled broadcast channel, i.e., our goal is to find broadcast schedules that minimze overall energy consumption, where sending at speed s for t time units consumes energy $t \cdot s^\alpha$ and transmission speed for each broadcast needs to be fixed when it is started. The main part of the paper is focused on the case that the server holds only a single message and every request defines a strict deadline before which a full broadcast has to be performed. We present an $\mathcal{O}(2^\alpha)$-competitive deterministic online algorithm and prove that this is asymptotically best possible even allowing randomization. We then discuss some possible problem extensions. For the case of multiple different messages we prove that an extension of our online algorithm achieves competitive ratio $(4c-1)^\alpha$ if the lengths of requests do not vary by more than a factor of c. For the problem variation in which the speed of running broadcasts may be changed, we present lower bounds showing that competitive ratios that depend exponentially on α are still unavoidable.

1 Introduction

Classical objectives in online broadcasting usually abstract away from the precise hardware architecture of the underlying computing machinery. They mostly aim at producing solutions that ensure a high degree of convenience for the serviced clients, but do not take into account the cost of actually realizing the solution. While this approach is quite reasonable in many traditional scenarios, recent years have brought about an increasing number of applications in which these issues become non-neglectible.

The most important factor determining the *cost* of running a broadcasting algorithm in practical applications is the algorithm's energy consumption. In fact, energy efficiency has become a premier objective in many different areas of computer science and recent advances have already led to changes in the structure of processors, graphic cards and other parts of computer hardware. Reduced energy consumption offers new application areas for computer systems. Multi-agent systems consisting of dozens of small, self-sustaining units support engineers and researchers in an increasing number of applications that range from production process planning to geographical observations [6]. Multi-agent systems depend on a reliable communication link between them that is typically provided by means of a wireless connection. Due to characteristics of their operation areas they are likely to be small and, consequently, carry a limited power supply. To use this as efficiently as possible specialized hardware like, e.g., low-power CPUs are utilized.

However, the energy consumed by the wireless connection is also far from being negligible. As wireless communication is implicitly done via broadcasts we propose to exploit this fact. We focus on a single agent that acts as a data server and adapt the situation introduced by Yao et al. in their seminal work [14]: we consider requests that have individual release times and deadlines and allow that multiple requests for the same piece of data can be answered by a single broadcast. While doing this results in a smaller number of

*Dept. of Computer Science, Dortmund University, Otto-Hahn-Str. 14, 44221 Dortmund, Germany. E-mail: patrick.briest@cs.uni-dortmund.de. Supported by DFG grant Kr 2332/1-2 within Emmy Noether program.

†Dept. of Computer Science, Freiburg University, Georges-Köhler-Allee 79, 79110 Freiburg, Germany. E-mail: gunia@informatik.uni-freiburg.de. Supported by DFG research training program No 1103 'Embedded Microsystems'.

broadcasts needed to answer all requests, it also reduces the time slot left for the broadcast at hand and, thus, requires higher transmission speed.

In previous works (e. g., [4] and [9]) the transmission power is merely used to adjust the transmission range in order to construct a topology that supports broadcasts but minimizes the energy consumption. We propose a completely different usage of the transmission power and use it to adjust the maximal transmission speed of the broadcast channel. As observed for example at the 802.11b-WLAN technology, the signal-to-noise ratio needed to send a transmission increases with increasing transmission speed [12] and, thus, higher speed results in increased energy consumption. Looking at it from the optimistic point of view, the server can reduce its energy consumption by simply keeping transmission speed low. We assume here that at speed s the power consumption is s^α per time unit, where $\alpha \geq 2$ is constant.

1.1 Related Work

Extensive research on various versions of online broadcasting has been going on for several years. The most popular problem variation aims at flowtime minimization, i.e., minimizing the time span between the arriving of a request and the time by which it is answered [1, 5]. Other objectives that have been investigated include minimization of the number of unsatisfied requests [8] or different QoS-measures allowing messages to be split up into an arbitrary number of smaller objects [11]. The question of energy efficiency in broadcasting is addressed in [7], where the main objective is still flowtime minimization, but an additional constraint defines the maximum energy consumption allowed for servicing a given sequence of requests. Adjusting the transmission energy has been used to reduce interference between different stations in large wireless networks by Burkhart et al. [3] and by Moscibroda et al. [10].

A related problem that has received a lot of attention also from the energy perspective is job scheduling with deadlines [14]. Here, a sequence of jobs, each with release time, deadline and a certain workload need to be scheduled on a single processor, such that all jobs are finished in time and the overall energy consumption is minimized. Here, the machine is speed-controlled and s^α for a constant $\alpha \geq 2$ represents the energy consumed per time unit at speed s. Yao et al. [14] present an algorithm that runs in polynomial time and computes an optimal schedule. They also propose two online algorithms AVERAGE RATE and OPTIMAL AVAILABLE and prove that the competitive ratio of AVERAGE RATE is somewhere between α^α and $2^\alpha\alpha^\alpha$. Bansal et al. [2] show that the competitive ratio of OPTIMAL AVAILABLE is exactly α^α. Furthermore, they prove that the competitive ratio of any online algorithm is at least $\Omega((4/3)^\alpha)$.

1.2 Preliminaries

As the base problem of this paper we consider a server that is confronted with a sequence $R = (r_1, r_2, \ldots)$ of requests for the same piece of information. This piece of information has a transfer volume of one, i.e., it can be broadcasted completely in $1/s$ time units at speed s. Request $r_j = (t_j, d_j)$ is posed at its release time t_j and has to be answered until its deadline d_j, i.e., the server's message has to be broadcasted completely between times t_j and d_j at least once. A broadcast performed at speed s for t time units consumes $t \cdot s^\alpha$ energy units for $\alpha \geq 2$. Therefore broadcasting the message completely in time t at fixed speed $1/t$ consumes $(1/t)^{\alpha-1}$ energy units. Due to the convexity of the energy function, it is not difficult to see that this is the minimal amount of energy needed to deliver the whole information within t time units. For the first part of this paper this will also be the only allowed type of broadcast, i.e., we will assume that the transmission speed for each broadcast needs to be fixed the moment it is started and cannot be changed while the broadcast is running. We consider the problem of finding a feasible schedule of broadcasts (i.e., answering all requests within their deadlines) that minimizes the overall energy consumption.

We will also consider two extensions of the problem defined above. First, we will investigate the case in which the server holds a larger number $k \in \mathbb{N}$ of messages. Every request $r_j = (t_j, d_j, m_j)$ then asks for a single message m_j to be delivered before its deadline. We then turn to the variation in which the speed of a running broadcast can be adapted by the algorithm. As before, sending at speed s for t time units causes an energy consumption of $t \cdot s^\alpha$.

Finally, let us introduce some notation that will be used throughout the rest of the paper. Given a sequence of requests R, we let $B = (b_1, b_2, \dots)$ and $B^* = (b_1^*, b_2^*, \dots)$ denote the corresponding sequences of broadcasts sent by an an online strategy or the optimal offline strategy, respectively. Sometimes it will be convenient to associate a broadcast $b_i = (s_i, f_i)$ with the interval $[s_i, f_i]$ defined by its starting and finishing times. For requests r_j as well as for broadcasts b_i we let $|r_j|$ and $|b_i|$ refer to their lengths.

1.3 Contributions

To the authors' best knowledge, this is the first analysis directly addressed to the minimization of energy consumption for broadcasting by speed scaling. We start by considering the restricted version of the problem in which the server holds only a single message and transmission speed cannot be changed while a broadcast is being performed. We first point out how to compute in polynomial time an optimal solution in the offline setting by applying an appropriate dynamic programming approach based on an interesting structural property of the optimal solution, which turns out to be separable into disjoint blocks of uniformly distributed broadcasts. We then present an easy to implement online algorithm and prove that it achieves competitive ratio $\mathcal{O}(2^\alpha)$. For the analysis we again utilize the structural results from the previous section, which yields a nice way of relating the cost incurred by our online strategy to the cost of an optimal offline solution. From a technical perspective, the advantage of this approach is the fact that separating the optimal solution into disjoint blocks allows an almost tight analysis with relatively small constants inside the $\mathcal{O}$-notation. These results are found in Sections 2.1 and 2.2.

It turns out that our algorithm's competitive ratio is best possible, as we proceed by showing a matching lower bound that holds even for randomized online algorithms. The lower bound is based on what could be called a *growing gap* argument. Towards a contradiction, we assume that a given online algorithm achieves a better competitive ratio and then construct a series of requests, such that in each newly added request there is a *gap*, i.e., a time interval that the algorithm cannot use for answering the last request. The key ingredient of the proof is a way of constructing the sequence, such that the size of the gap increases with every newly added request. The necessary technique is first developed for deterministic algorithms only. Adding randomness to the construction and applying some additional technical arguments, we then transfer the result to randomized algorithms, as well. Details are found in Section 2.3.

Since from a technical point of view it appears that good lower bounds require sequences of requests of heavily decreasing lengths, we also have a closer look at instances in which all requests are of identical length. We prove that in this case the competitive ratio of our algorithm improves to $(3/2)^\alpha$.

We point out that even this rather restricted single-message scenario is of practical interest. In mobile multi agent scenarios for example, a single agent might be broadcasting some current sensor status to the other agents whenever it is needed. In our model, this means that the server is in fact holding only a single message, although it might of course be changing over time.

We nevertheless continue by investigating the multiple-message scenario, in which the server holds any larger number $k \in \mathbb{N}$ of different messages. We present an extension of our online algorithm for the single-message case and show that it has competitive ratio $(4c-1)^\alpha$ if all requests have length between ℓ and $c\ell$ for some positive constants c and ℓ, i.e., if lengths vary by at most a factor of c. Especially, this yields a 3^α-competitive algorithm for requests of identical length. Surprisingly, these ratios do not depend on how many different messages are hosted by the server, i.e., they are completely independent from parameter k. Finally, we take a look at the effect of allowing the algorithm to adapt the speed of running broadcasts. We prove a lower bound of 1.38^α on the competitive ratio of any online algorithm in the general case and a lower bound of 1.09^α for requests of identical length. For results on these extensions see Section 3.

2 Single-Message Broadcasting

In this section we describe the results about the scenario in which the server holds a single message and transmission speed for a broadcast needs to be fixed at its beginning. We start by stating some facts about the optimal offline solution and develop a dynamic programming based algorithm in Section 2.1. We then prove our main result by presenting an $\mathcal{O}(2^\alpha)$-competitive online algorithm in Section 2.2 and proving its optimality by a matching lower bound in Section 2.3.

2.1 Computing the Offline Solution

Let $R = (r_1, \ldots, r_n)$ be a given sequence of requests and assume that an optimal broadcast schedule is given by $B^* = (b_1^*, \ldots, b_m^*)$, where the b_i^* are ordered chronologically. We call the release times and deadlines of requests in R *event points* and denote them by $E = (e_1, \ldots, e_{2n})$ again assuming chronological order. By $B(i, j, m)$, $i < j$, we refer to a non-interrupted block of m broadcasts of identical length with the first one starting at time e_i, the last one finishing at time e_j. The following lemma states that B^* is just a collection of blocks of this type. A proof is found in the appendix.

Lemma 1 *An optimal schedule for a given input instance consists of blocks $B(i_k, j_k, m_k)$ for $k = 1, 2, \ldots, \nu$.*

Based on this we solve the problem by a dynamic programming approach. Let T be the *table of overlaps*, i.e., $T(j)^{\text{first}}$ is the first request (the one with earliest release time) that contains event point e_j, $T(j)^{\text{width}}$ the number of requests that contain e_j. Let $\Delta = \max_j T(j)^{\text{width}}$. It is easy to check that it is possible to compute table T for a given input instance within time $O(n \log n)$ on space $O(n)$.

The tuple (i, j, k, l) denotes the subproblem in which only the requests $T(j)^{\text{first}} + i, T(j)^{\text{first}} + i + 1, \ldots, T(k)^{\text{first}} + l$ have to be answered and all broadcasts have to be performed within $[e_i, e_k]$. Slightly abusing notation the tuple also denotes the value of an optimal solution to this problem. Hence, we are interested in computing $(1, 0, 2n, T(2n)^{\text{width}} - 1)$.

The main idea of algorithm OFFLINE is to identify the blocks as found in Lemma 1 by dynamic programming. After initializing the table (i, j, k, l) with zeros, it already contains the correct solutions for $i = k$ and $j = l$ as there is either no time to perform a broadcast or no request to answer. OFFLINE then successively computes the optimal solutions for (i, j, k, l) in lines 3 to 6. By Lemma 1 it follows that an optimal solution to (i, j, k, l) must either be identical to block $B(j, k, m)$ for some $m \geq 1$, or must contain some event point e_i at which the problem can be split up, since it is not overlapped by any of the optimal broadcasts.

Input: Requests with release times and deadlines
Result: Minimal costs of a schedule of broadcasts answering them

1 Fill up table of overlaps, i. e., $T(i)^{\text{first}}$ and $T(i)^{\text{width}}$ for $1 \leq i \leq 2n$
2 Initialize table of optimal solutions, i. e., $(i, j, k, l) \leftarrow 0$ for $1 \leq i, k \leq 2n, 1 \leq j, k \leq \Delta$
3 **for** increasing $k - i$ and $l - j$ do **do**
4 $\quad v_1 \leftarrow \min\{(i, j, i', j') + (i', j', k, l) \mid i \leq i' \leq k \wedge j \leq j' \leq j\}$
5 $\quad v_2 \leftarrow \min\{B(i, k, m) \mid 1 \leq m \leq n\}$
6 $\quad (i, j, k, l) \leftarrow \min\{v_1, v_2\}$
end
7 **return** $(1, 0, 2n, T(2n)^{\text{width}} - 1)$

Algorithm 1: OFFLINE.

Theorem 1 *For a given input instance algorithm* OFFLINE *computes an optimal schedule in time $\mathcal{O}(\Delta^2 \cdot n^4)$ and on space $\mathcal{O}(\Delta^2 \cdot n^2)$.*

2.2 An Online Algorithm

Algorithm ONLINE-SM proceeds as follows. If a request $r = (t, d)$ arrives at time t while the channel is idle, we start a broadcast that uses the full length $d - t$ of the request. If the channel is busy and the currently running broadcast is scheduled to finish at time τ, we abort and start a new broadcast if at least half of the interval $[t, d]$ defined by r lies before τ. In the implementation below τ denotes the end of the currently running broadcast, ρ refers to the earliest deadline of any request that needs to be answered by a broadcast starting after τ.

```
1  τ ← +∞, ρ ← +∞
2  if a request r = (t, d) arrives then
3      if channel is idle then
4          τ ← d
5          start broadcast at speed (τ − t)^{−1}
6      if channel is busy then
7          if τ − t ≥ d − τ then
8              abort current broadcast
9              τ ← min{τ, d}
10             ρ ← +∞
11             start broadcast at speed (τ − t)^{−1}
12         else
13             ρ ← min{ρ, d}
14 if a broadcast finishes and ρ < +∞ then
15     τ ← ρ
16     ρ ← +∞
17     start broadcast at speed (τ − t)^{−1}
```

Algorithm 2: ONLINE-SM.

Theorem 2 *Let E_{ON} denote the energy consumption of algorithm* ONLINE-SM *on any sequence of requests, E_{OPT} the value of an optimal offline solution on the same sequence. It holds that*

$$E_{ON} \leq \left(\frac{\alpha}{\alpha - 1}\right)^2 2^\alpha \cdot E_{OPT}.$$

Before presenting the proof of Theorem 2 we point out that a better competitive ratio is obtained if we require all requests to have identical length.

Theorem 3 *Let E_{ON} denote the energy consumption of algorithm* ONLINE-SM *on any sequence of requests of identical length, E_{OPT} the value of an optimal offline solution on the same sequence. It holds that*

$$E_{ON} \leq \frac{2\alpha}{\alpha - 1}\left(\frac{3}{2}\right)^\alpha \cdot E_{OPT}.$$

We proceed by proving Theorem 2. First observe that algorithm ONLINE-SM outputs a feasible solution, i.e., every request is indeed answered by one of the algorithm's broadcasts. Consider a single request $r = (t, d)$ arriving at time t. If either line 3 or line 7 apply, the algorithm immediately starts a broadcast with finishing time $\tau \leq d$. Afterwards this broadcast may well be aborted due to newly arriving requests. However, eventually there must be a completed broadcast finishing before τ and, thus, answering r. If neither line 3 nor line 7 apply, we get $\tau < \rho \leq d$ due to line 13 of the algorithm. If the current broadcast is completed at time τ, the algorithm starts a broadcast with finishing time $\rho \leq d$ due to lines 14 to 17, which answers r. If the current broadcast is aborted due to other requests, the first completed broadcast with finishing time before τ will answer r. This completes the proof of correctness.

In order to prove the competitive ratio claimed above we first need to bound the cost incurred by our algorithm due to aborted broadcasts. The following lemma states that this cost becomes negligible for larger values of α. Due to space limitations the proofs of Lemma 2 and Theorem 3 are omitted.

Lemma 2 *Let E_{ab} denote the energy consumed by algorithm* ONLINE-SM *due to aborted broadcasts, E_{co} the energy used by completed broadcasts. Then $E_{ab} \leq (1/(\alpha-1))E_{co}$ for any given sequence of requests.*

Proof of Theorem 2: For the remainder of the proof it will be important to know to which request to assign the cost of a single broadcast. Given broadcast b sent by algorithm ONLINE-SM we say that b *is linked to* request r, if b is started due to r in lines 5, 11 or 17. It is straightforward to observe that $|b| \geq |r|/2$, whenever b is linked to r.

Consider the optimal offline solution. By Lemma 1 broadcasts B sent by the offline strategy can be partitioned into groups $B_1, \ldots, B_\ell$, such that each B_i is a non-interrupted sequence of broadcasts of identical length. Let us fix some group B_i containing some number $k \geq 1$ of broadcasts, with the first one starting at time s, the last one finishing at time t. Thus, every broadcast in B_i has length $(t-s)/k$.

Let R_i denote the set of requests answered by broadcasts in B_i in the offline setting. Clearly, all requests in R_i have length at least $(t-s)/k$. We are going to bound the energy E consumed by algorithm ONLINE-SM for sending broadcasts linked to requests in R_i. To this end, let $E = E_1 + E_2 + E_3$, where E_1 denotes energy consumption before time s, E_2 and E_3 refer to energy consumed between times s and t and after t, respectively.

Let us first consider E_1. We know that every request in R_i has a deadline no earlier than $s + (t-s)/k$, as otherwise it could not be answered by a broadcast from B_i. Now assume that algorithm ONLINE-SM is sending a broadcast b linked to some request $r \in R_i$ at time $u < s$. It follows that $|b| \geq (1/2)|r| \geq (1/2)(s + (t-s)/k - u)$ and, thus, the algorithm is sending at speed $|b|^{-1} \leq 2(s+(t-s)/k-u)^{-1}$ at time u. We can then write that

$$E_1 \leq 2^\alpha \int_0^s (s + \frac{t-s}{k} - u)^{-\alpha} du = \frac{2^\alpha}{\alpha-1} u^{-\alpha+1} \Big|_{s+\frac{t-s}{k}}^{\frac{t-s}{k}} \leq \frac{2^\alpha}{\alpha-1}\left(\frac{t-s}{k}\right)^{-\alpha+1}.$$

We next consider E_2 and E_3. As observed before, broadcasts linked to requests in R_i have length at least $(t-s)/(2k)$. Hence, no more than $2k$ such broadcasts can be started between times s and t. Broadcasts starting after time t are sent only if requests from R_i have been left unanswered. First observe that at most one such broadcast is sent, as it will clearly answer all requests from R_i, which we know have starting times before t.

We then observe that requests from R_i can only be left unanswered before time t, if at most $2k-2$ broadcasts starting between s and t are sent. To see this, note that every $r \in R_i$ is posed before time $t - (t-s)/k$, since otherwise it could not be answered by broadcasts in B_i. However, since broadcasts of algorithm ONLINE-SM linked to requests in R_i have length at least $(t-s)/(2k)$, no more than $2k-2$ such broadcasts can be started between s and $t-(t-s)/k$. It follows that

$$E_2 + E_3 \leq 2k\left(\frac{t-s}{2k}\right)^{-\alpha+1} = 2^\alpha k \left(\frac{t-s}{k}\right)^{-\alpha+1}.$$

On the other hand, the energy consumed by the offline strategy answering requests from R_i is $E^* = k \cdot ((t-s)/k)^{-\alpha+1}$, since k broadcasts of length $(t-s)/k$ are sent. Finally, we obtain

$$E \leq \left(\frac{2^\alpha}{\alpha-1} + 2^\alpha k\right)\left(\frac{t-s}{k}\right)^{-\alpha+1} \leq \frac{\alpha}{\alpha-1} 2^\alpha k \left(\frac{t-s}{k}\right)^{-\alpha+1} = \frac{\alpha}{\alpha-1} 2^\alpha \cdot E^*.$$

Let E_{ON} and E_{OPT} denote the energy consumed by algorithm ONLINE-SM and the optimal offline strategy on the complete set of requests, respectively. Summing up over all groups R_i and taking into account energy consumption due to aborted broadcasts, we get

$$E_{ON} \leq \left(\frac{\alpha}{\alpha-1}\right)^2 2^\alpha \cdot E_{OPT},$$

which proves the theorem. □

2.3 A Lower Bound

In this section we will show that algorithm ONLINE-SM is asymptotically best possible. We start with a matching lower bound for deterministic algorithms in Theorem 4. Theorem 5 extends this to randomized algorithms.

Theorem 4 *The competitive ratio of every deterministic online algorithm for single-message broadcasting is* $\omega((2-\varepsilon)^\alpha)$ *for any* $\varepsilon > 0$.

Here is the high-level idea of the proof: We construct a sequence of requests of exponentially decreasing lengths. In every step of the construction, there will be a time interval contained within the very last request, which the algorithm cannot use for sending a broadcast answering it. We will refer to this prohibited time interval as a *gap*. The key ingredient of the proof is Lemma 3, which states that if an online algorithm has competitive ratio $\mathcal{O}((2-\varepsilon)^\alpha)$ for some $\varepsilon > 0$, the relative length of the gap (i.e., compared to the length of the last request) increases steadily, forcing the algorithm to violate its competitive ratio after a number of requests.

Proof of Theorem 4: Towards a contradiction assume that deterministic online algorithm A has competitive ratio $\mathcal{O}((2-\varepsilon)^\alpha)$ for some constant $\varepsilon > 0$. We denote our sequence of requests as $r_0, r_1, \ldots$ and let $R_j = (r_0, \ldots, r_j)$. Since A is deterministic, we can construct the input sequence in step by step manner, i.e., we can define request r_{j+1} depending on the algorithm's observed behavior on R_j.

Before we give a detailed description of our construction, we need to define the notion of a *gap* more formally. We have to consider two different types of gaps. We say that r_j has a gap of relative length δ *at its beginning*, if a broadcast that has been started before r_j is posed is not aborted and finishes $\delta \cdot |r_j|$ time units after r_j is posed. Thus, the broadcast answering r_j must be started at least $\delta \cdot |r_j|$ time units after the request is actually posed. On the other hand, we say that r_j has a gap of relative length δ *at its end*, if there exists a request r_i, $i < j$, that has a deadline $\delta \cdot |r_j|$ time units before the deadline of r_j and needs to be answered by the same broadcast as r_j. In this situation, the broadcast answering r_j clearly needs to finish $\delta \cdot |r_j|$ time units before its deadline, as otherwise r_i were left unanswered. Gap positions are depicted in Figure 1.

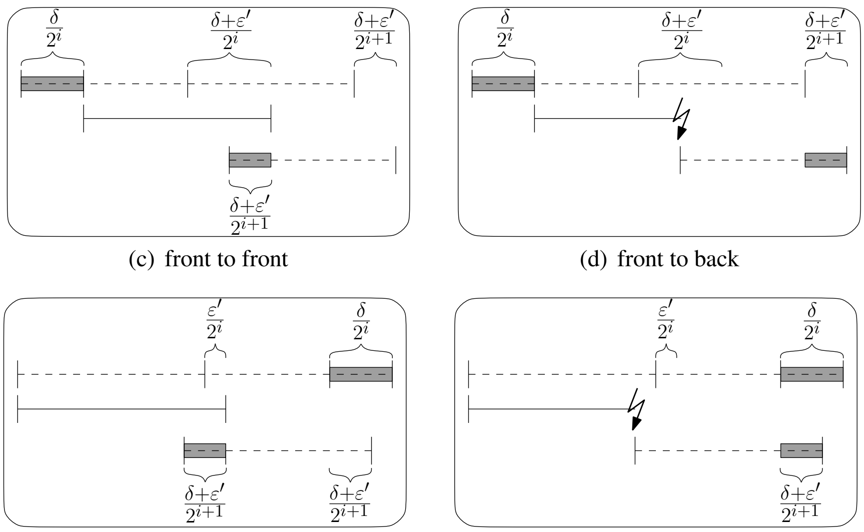
Let now $E_{OPT}(R_j)$ denote the cost of an optimal offline solution on R_j and assume for the moment that $E_{OPT} = \mathcal{O}(|r_j|^{-\alpha+1})$, i.e., assume that the cost is dominated by the length of the last request. Having this it is clear that algorithm A's broadcast answering r_j must have length at least $(1/2 + \varepsilon')|r_j|$ for some appropriately chosen $\varepsilon' > 0$ in order to guarantee its competitive ratio. Let now $r_j = (t_j, d_j)$ and $m_j = (t_j + d_j)/2$ refer to the middle of the interval defined by r_j. We set $r_0 = (0, 1)$ and say that r_0 has a gap of relative length 0 at its beginning. For the definition of r_{j+1} we distinguish two cases. If r_j has a gap of relative length δ at its beginning, we set $r_{j+1} = (m_j + (\delta + \varepsilon')|r_{j+1}|, d_j + (\delta + \varepsilon')|r_{j+1}|)$. If a gap of the same length is at the end of r_j, we set $r_{j+1} = (m_j - (\delta - \varepsilon')|r_{j+1}|, d_j - (\delta - \varepsilon')|r_{j+1}|)$. Intuitively, r_{j+1} has length 2^{-j-1} spanning the second half of request r_j and is shifted according to the gap's position in r_j.

By Lemma 3 the relative length of the gap increases by at least ε' in each step. Hence, its relative length exceeds $1/2$ at some point and there must exist a request r_n that algorithm A answers by a broadcast of length at most $(1/2)|r_n|$. Thus, the energy consumption of algorithm A on R_n is $E_A(R_n) = \Omega(2^{(\alpha-1)(n+1)})$.

It remains to bound $E_{OPT}(r_n)$ from above. The optimal schedule uses the full length of $|r_n|$ for its last broadcast. Consider then an arbitrary request r_j. If r_j contains some r_k, $k > j$, then it is answered by the broadcast answering r_k. Otherwise, all requests r_k, $k > j$, are posed after m_j and r_j can be answered by a broadcast of length $m_j - t_j = |r_j|/2$. Now remember that $|r_j| = 2^{-j}$ and we obtain that

$$E_{OPT}(R_n) \leq \sum_{j=0}^{n-1} (|r_j|/2)^{-\alpha+1} + |r_n|^{-\alpha+1} = \sum_{j=0}^{n-1} 2^{(\alpha-1)(j+1)} + 2^{(\alpha-1)n} = \mathcal{O}(2^{(\alpha-1)n}),$$

which finishes the proof. □



(c) front to front (d) front to back

(e) back to front (f) back to back

Figure 1: Construction for a gap of relative length δ with respect to its position in the i-th request. Intervals corresponding to requests and broadcasts are depicted as dashed and solid lines, respectively. The flash symbol denotes abortion of a running broadcast. Boxes mark gaps as described in Theorem 4.

Lemma 3 *Given an online algorithm as in the proof of Theorem 4, the relative length of the gap increases with each newly added request by at least ε'.*

Proof: Due to the regular construction of the instance it suffices to show the claim for a single step. We have a look at requests r_i and r_{i+1}. To simplify notation, assume w.l.o.g. that the length of request r_i is 1. Hence, $|r_{i+1}| = 1/2$ and the length of the broadcast that answers request r_i has to be of length at least $(1/2 + \varepsilon')$ because of algorithm A's competitive ratio. It follows that due to our construction this broadcast is still being performed the moment request r_{i+1} is issued. As the algorithm is not allowed to change the running broadcasts's speed, it has only two ways to deal with this situation: it either aborts or continues the broadcast answering r_i. We do a case inspection on the position of the gap in r_i and the algorithm's behavior. All cases are depicted in Figure 1. Let δ denote the relative length of the gap in r_i.

Case (a) As the algorithm does not abort the broadcast it does not finish before time $1/2 + \delta + \varepsilon'$. Remember that the request r_{i+1} is issued at time $(1 + \delta + \varepsilon')/2$. Hence, at least the first $(\delta + \varepsilon')/2$ time units of request r_{i+1} are not used to answer it. It follows that the relative length of its gap is at least $(\delta + \varepsilon')$.

Case (b) The algorithm aborts the broadcast instantly the moment request r_{i+1} is issued. Hence, the next broadcast must not finish beyond time 1. As request r_{i+1} is issued at time $(1 + \delta + \varepsilon')/2$ and has length $1/2$, its deadline is at time $1 + (\delta + \varepsilon')/2$. This yields a gap of relative length $(\delta + \varepsilon')$ at its end. The same is obviously true if the algorithm aborts the running broadcast not immediately, but at some later point of time.

Case (c) The broadcast that answers request r_i is scheduled to finish at time $1/2 + \varepsilon'$ at the earliest. Due to the construction request r_{i+1} is issued at time $(1 - \delta + \varepsilon')/2$. Since the algorithm does not abort the broadcast prematurely, the relative length of the gap of request r_{i+1} is at least $\delta + \varepsilon'$.

Case (b) The algorithm aborts the broadcast the moment request r_{i+1} is issued. Remember that there is a gap of relative length δ at the end of request r_i. Hence, there exists a request with deadline $1 - \delta$. This request is still unanswered, since the broadcast was aborted. Therefore, the next broadcast must not finish beyond this time. As request r_{i+1} is issued at time $(1 - \delta + \varepsilon')/2$ and has length $1/2$, its deadline is at time $1 - (\delta - \varepsilon')/2$. We conclude that this yields a gap of relative length at least $\delta + \varepsilon'$ at the end of this request. Again, the same argumentation holds if the broadcast is aborted at a later point. $\square$

The next theorem extends this result to randomized algorithms as well. We briefly sketch the key idea. As the length of the broadcast answering the last request dominates the cost of the solution, we know that the expected length of the broadcast answering r_j must not fall below $(1/2 + \varepsilon')|r_j|$. Assuming that each request is added as before, we obtain that the expected length of the gap increases in every step. However, as an adversary we do not know the random coin flips of the algorithm and, thus, do not know the position of the gap. We solve this problem by shifting requests in either of the two possible directions randomly. The key observation is that for constant $\varepsilon' > 0$ we need only a constant number of successful steps to reach a sufficient gap size and, thus, this randomized construction is still strong enough to achieve a contradiction.

Theorem 5 *The expected competitive ratio of every (randomized) online algorithm for single-message broadcasting is $\omega((2-\varepsilon)^\alpha)$ for any $\varepsilon > 0$.*

3 Extensions

Finally, we will briefly sketch a number of results for some natural extensions of our problem. Section 3.1 presents a competitive online algorithm for the case that more than a single message needs to be broadcasted. Section 3.2 discusses the implications of allowing the speed of a running broadcast being changed.

3.1 An Online Algorithm for Multiple Messages

We present an online algorithm for the case that the server holds some number $k \in \mathbb{N}$ of different messages. Request $r = (t, d, m)$ now also specifies the message m that needs to be received. We assume that requests have lengths that vary between ℓ and $c\ell$ for some positive constants c and ℓ. Algorithm ONLINE-MM repeatedly collects a number of requests which are then answered by a sequence of broadcasts of overall length $\ell/2$. In the implementation below N denotes the set of messages that need to be sent within the next sequence of broadcasts, L refers to messages that will be sent later. Parameters τ and ρ are the finishing times of the next two upcoming broadcasting sequences.

$N, L \leftarrow \emptyset$, $\tau, \rho \leftarrow +\infty$

if *a request* $r = (t, d, m)$ *arrives* **then**
    if *channel is idle* **then**
        $\tau \leftarrow \min\{\tau, d - \ell/2\}$
        $N \leftarrow N \cup \{m\}$
    if *channel is busy* **then**
        $\rho \leftarrow \min\{\rho, d - \ell/2\}$
        $L \leftarrow L \cup \{m\}$

at *time* τ **do**
    start broadcasts of length $\ell/(2|N|)$ each for messages in N
    $N \leftarrow \emptyset$, $\tau \leftarrow +\infty$

if *a sequence of broadcasts finishes and* $\rho < +\infty$ **then**
    $N \leftarrow L$, $L \leftarrow \emptyset$
    $\tau \leftarrow \rho$, $\rho \leftarrow +\infty$

Algorithm 3: ONLINE-MM.

The following theorem states that algorithm ONLINE-MM has a constant competitive ratio that depends only on c, i.e., the factor by which request lengths may vary. We specifically note that this competitive ratio is independent from the number k of messages held by the server. A proof is found in the Appendix.

Theorem 6 *Let E_{MM} denote the energy consumption of algorithm* ONLINE-MM *on any sequence of requests with lengths varying between ℓ and $c\ell$ for some fixed $c \geq 1$, E_{OPT} the value of an optimal offline solution on the same sequence. It holds that $E_{MM} \leq (4c-1)^\alpha \cdot E_{OPT}$.*

3.2 More Flexible Speed-Adjustment

We will briefly address another extension to our problem. It is conceivable to allow the server a speed change during a broadcast without enforcing a restart. Obviously, the lower bound presented earlier does not apply to this situation. The next theorem states that the competitive ratio of any algorithm capable of speed-up depends exponentially on α, as well. Its proof can be found in the appendix.

Theorem 7 *The competitive ratio of any (randomized) online algorithm capable of speed-up is* $\omega((\gamma - \varepsilon)^{\alpha})$ *for every* $\varepsilon > 0$*, where* $\gamma = (5 + 5\sqrt{5})/(5 + 3\sqrt{5}) > 1.38$*. If all requests are of identical length the competitive ratio is at least* $\Omega(1.09^{\alpha})$*.*

References

[1] N. Bansal, D. Coppersmith, and M. Sviridenko. Improved Approximation Algorithms for Broadcast Scheduling. In *Proc. of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2006.

[2] N. Bansal, T. Kimbrel, and K. Pruhs. Dynamic Speed Scaling to Manage Energy and Temperature. In *Proc. of the IEEE Symposium on Foundations of Computer Science (FOCS)*, 2004.

[3] M. Burkhart, P. von Rickenbach, R. Wattenhofer, and A. Zollinger. Does Topology Control Reduce Interference? In *Proc. of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2004.

[4] A. Clementi, P. Crescenzi, P. Penna, G. Rossi, and P. Vocca. On the Complexity of Computing Minimum Energy Consumption Broadcast subgraphs. In *Proc. of the Symposium on Theoretical Aspects of Computer Science (STACS)*, 2001.

[5] J. Edmonds and K. Pruhs. Multicast Pull Scheduling: When Fairness is Fine. *Algorithmica*, 36, 3:315–330, 2003.

[6] G.-Y. Gao and S.-K. Wang. A Multi-Agent System Architecture for Geographic Information Gathering. *Journal of Zhejiang University SCIENCE*, 5(11):1367–1373, 2004.

[7] C. Gunia. On Broadcast Scheduling with Limited Energy. In *Proc. of the Conference on Algorithms and Complexity (CIAC)*, 2006.

[8] B. Kalyanasundaram, K. Pruhs, and M. Velauthapillai. Scheduling Broadcasts in Wireless Networks. In *Proc. of the European Symposium on Algorithms (ESA)*, 2000.

[9] X-Y. Li, W-Z. Song, and W. Wang. A Unified Energy-Efficient Topology for Unicast and Broadcast. In *Proc. of the ACM-IEEE International Conference on Mobile Computing and Networking (MobiCom)*, 2005.

[10] T. Moscibroda and R. Wattenhofer. Minimizing Interference in Ad Hoc and Sensor Networks. In *Proc. of the Joint Workshop on Foundations of Mobile Computing*, 2005.

[11] K. Pruhs and P. Uthaisombut. A Comparison of Multicast Pull Models. In *Proc. of the European Symposium on Algorithms (ESA)*, 2002.

[12] D. Qiao, S. Choi, and K.-G. Shin. Goodput Analysis and Link Adaptation for IEEE 802.11a Wireless LANs. *IEEE Transactions on Mobile Computing*, 1(4):278–292, 2002.

[13] A.C.-C. Yao. Probabilistic Computations: Towards a Unified Measure of Complexity. In *Proc. of the IEEE Symposium on Foundations of Computer Science (FOCS'77)*, pages 222–227, 1977.

[14] F. Yao, A. Demers, and S. Shenker. A Scheduling Model for Reduced CPU Energy. In *Proc. of the IEEE Symposium on Foundations of Computer Science (FOCS)*, 1995.

A Appendix

Lemma 1. *An optimal schedule for a given input instance consists of blocks $B(i_k, j_k, m_k)$ for some i_k, j_k, m_k for $k = 1, 2, \dots, \nu$.*

Proof: For the sake of contradiction, we pick the largest block that has at most one border and does not match any release time or deadline except for its borders. As it is possible that a block consists of merely one broadcast such a block has to exist if this lemma is not fulfilled. It is not difficult to see that this block can collide with other blocks at most at its borders. For now, let us assume that this block has at most one border. Then we can further assume, without loss of generality, that there is no border at the end of the block. We scale the block up by slowing down its last broadcast by a sufficiently small amount. We see that the solution remains feasible but its cost decreases since slowing down a broadcast results in lower cost. This contradicts the fact that the schedule was optimal in the first place. In the same manner we can prove that all requests of one block have to have the same length if the block has two borders; otherwise we can reduce the cost by rescaling the broadcasts while keeping the block's size constant. In doing so the schedule remains feasibly since it matches no release time and deadlines except for its borders. □

Theorem 1. *For a given input instance algorithm* OFFLINEALG *computes an optimal schedule in time $\mathcal{O}(\Delta^2 \cdot n^4)$ and on space $\mathcal{O}(\Delta^2 \cdot n^2)$.*

Proof: The main idea of algorithm OFFLINEALG (Algorithm 1 on Page 4) is to divide the given problem in two smaller subproblems while ensuring that an optimal solution for the original problem can be composed of optimal solutions of the subproblems. We use the event points as cut-off points for the division. If we knew that at some time t a broadcast ends and requests $r_i, r_{i+1}, \dots, r_n$ are left unanswered, we would canonically obtain two subproblems:

- Create a schedule with minimal cost that answers requests $r_1, r_2, \dots, r_{i-1}$ and whose last broadcast ends at time t.
- Create a schedule with minimal cost that answers requests $r_i, r_{i+1}, \dots, r_n$ and whose first broadcast starts after time t.

Obviously, we can compose an optimal solution for the original problem of optimal solutions for these two subproblems. The problem is that we do not know when a broadcast ends. This is the point at which Lemma 1 comes in. Intuitively speaking, it tells us that in an optimal solution there is either a broadcast whose ending point coincides with an event point or the structure of the optimal solution is trivial.

Let T be the table of overlaps mentioned above, i. e., $T(j)^{\text{first}}$ is the first request that overlaps event point e_j and $T(j)^{\text{width}}$ the number of overlapping requests. In algorithm OFFLINEALG the tuple (i, j, k, l) denotes the subproblem in which solely the requests $T(j)^{\text{first}} + i, T(j)^{\text{first}} + i + 1, \dots, T(k)^{\text{first}} + l$ have to be answered and all broadcasts have to be executed within $[e_i, e_k]$. The tuple also denotes the value of an optimal solution to this problem. Hence, we are interested in $(1, 0, 2n, T(2n)^{\text{width}} - 1)$.

With this information at hand we have a look the pseudo code of algorithm OFFLINEALG. After initializing the table T of overlaps, it creates a table (i, j, k, l) that is intended to contain the values of the subproblems defined above. The algorithm initializes this table with zeros, that is the correct value for all subproblems with $i = k$ or $j = l$ as there is either no time for executing a broadcast or no requests to answer. Beginning with line 3 it starts filling up the table by increasing $k - i$ and $l - j$. Within these lines v_1 represents the minimal cost of the "trivial solution" and v_2 the minimal cost of a solution obtained by solving subproblems. Finally, in line 7 the value of an optimal solution is returned.

Obviously, space $\mathcal{O}(n)$ suffices to store table T. Furthermore, the table (i, j, k, l) does not consume more than $\mathcal{O}(\Delta^2 \cdot n^2)$ space since each entry consists of merely one value. This proves the claimed upper bound on the space consumption. As it is easy to verify that lines 4 to 6 can be executed in time $\mathcal{O}(n^2)$ the upper bound on the running time holds as well.

Finally, we will prove the correctness of algorithm OFFLINEALG. According to Lemma 1, an arbitrary optimal solution consists of blocks of broadcasts. The broadcasts within one block have all the same length. The starting and ending point of each of these blocks coincides with an event point. Therefore, each block is represented by a specific subproblem (i, j, k, l) with appropriately chosen parameters i, j, k, l. We point out that this subproblem contains just one broadcast block in its optimal solution. As a consequence, the algorithm will compute the optimal solution of this block in line 6. We use this as an induction basis for an induction over the number of broadcast blocks.

If an optimal solution of a subproblem (i, j, k, l) consists of more than one broadcast block, the endpoint of one broadcast block has to coincide with an event point due to Lemma 1. Hence, the subproblem can be divided into two subproblems (i, j, i', j') and (i', j', k, l) with appropriately chosen i' and j'. Since the algorithm traverses all feasible cut-off points it will certainly try the "right one". Each of the resulting subproblems contains less broadcast blocks than (i, j, k, l) and, therefore, an optimal solution for each of them is already known to OFFLINEALG. Hence, the corresponding parameters will yield the lowest cost and the optimal solutions for the subproblem (i, j, k, l) will be constructed in line 4. □

Lemma 2. *Let E_{ab} denote the energy consumed by algorithm* ONLINE-SM *due to aborted broadcasts, E_{co} the energy used by completed broadcasts. Then*

$$E_{ab} \leq \frac{1}{\alpha - 1} E_{co}$$

for any given sequence of requests.

Proof: Let $B = (b_1, \ldots, b_n)$ be a sequence of (aborted and completed) broadcasts sent by algorithm ONLINE-SM on some sequence of requests and assume that broadcasts are ordered by their starting times. We can partition B into groups $B_i = (b_k, \ldots, b_\ell)$, such that in each group broadcasts $b_k, \ldots, b_{\ell-1}$ are aborted, broadcast b_ℓ is completed.

Let us now fix some group $B_i = (b_k, \ldots, b_\ell)$ and assume w.l.o.g. that the first broadcast b_k starts at time 0. Furthermore, let b_ℓ start at time s and finish at time t. In algorithm ONLINE-SM every starting broadcast is scheduled to be finished at time τ for the current value of τ. During our subsequence of broadcasts the channel is never idle and, while this is the case, the value of τ can only be decreased. It follows that $b_k, \ldots, b_{\ell-1}$ are all scheduled to be finished at some time $\tau \geq t$. Thus, at any time $0 \leq u \leq s$ our algorithm is sending at speed at most $(t-u)^{-1}$, which bounds energy consumption at this point of time by $(t-u)^{-\alpha}$. Let $E_{ab}(B_i)$ and $E_{co}(B_i)$ denote the energy consumption on broadcasts B_i alone. From the above we conclude that

$$E_{ab}(B_i) \leq \int_0^s (s-u)^{-\alpha} du = \frac{1}{\alpha - 1} u^{-\alpha+1} \Big|_t^{t-s} \leq \frac{1}{\alpha - 1} (t-s)^{-\alpha+1}.$$

On the other hand, b_ℓ is sent at speed $(t-s)^{-1}$ and, thus, $E_{co}(B_i) = (t-s)^{-\alpha+1}$. Summing up over all groups gives the claim. An illustration of the proof is found in Fig. 2. □

Theorem 3. *Let E_{ON} denote the energy consumption of algorithm* ONLINE-SM *on any sequence of requests of identical length, E_{OPT} the value of an optimal offline solution on the same sequence. It holds that*

$$E_{ON} \leq \frac{2\alpha}{\alpha - 1} \left(\frac{3}{2}\right)^{\alpha} \cdot E_{OPT}.$$

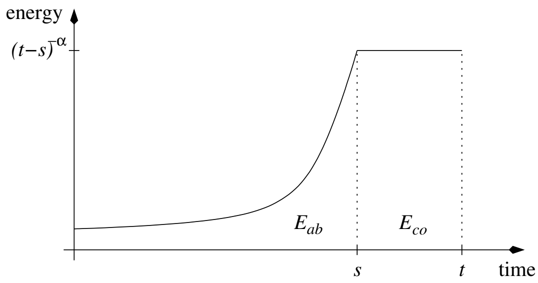


Figure 2: Energy consumption due to aborted broadcasts.

Proof: Let R be a given sequence of requests and assume w.l.o.g. that all requests have length 1. Furthermore, let B and B^* denote the broadcasts sent by algorithm ONLINE-SM and the optimal offline strategy, respectively. By the definition of algorithm ONLINE-SM, every $b_i \in B$ starts at the release time or deadline of some request $r_i^1 \in R$ and ends at the deadline of some request $r_i^2 \in R$, where not necessarily $r_i^1 \neq r_i^2$. We say that b_i is *flanked* by $r_i^1 = (t_i^1, d_i^1)$ and $r_i^2 = (t_i^2, d_i^2)$ and distinguish the cases that (1) $b_i = (t_i^2, d_i^1)$ or (2) $b_i = (d_i^1, d_i^2)$ as depicted in Fig. 3.

Let now $\Delta_i = d_i^1 - t_i^2$ denote the *overlap* of r_i^1 and r_i^2. Since all requests have length 1, we know that $|b_i| \geq 1/2$ and, thus, $\Delta_i \geq |b_i| \geq 1/2$ in case (1), $\Delta_i \leq 1 - |b_i| \leq 1/2$ in case (2). We next lower bound the cost incurred by the offline strategy answering r_i^1 and r_i^2.

In case (1) we have that $|b_i| = \Delta_i$. The optimal schedule may either answer both r_i^1 and r_i^2 with a single broadcast, or it has to send at least two broadcasts between times t_i^1 and d_i^2, where $d_i^2 - t_i^1 = 2 - \Delta_i$. In any case the optimal strategy sends at least one broadcast b_i^* of length

$$|b_i^*| \leq \max\left\{\Delta_i, 1 - \frac{\Delta_i}{2}\right\} \leq \max\left\{|b_i|, \frac{3}{4}\right\} \leq \frac{3}{2}|b_i|,$$

answering one of the flanking requests r_i^1 and r_i^2.

In case (2) we have $|b_i| = 1 - \Delta_i$ and a broadcast b_i^* sent by the offline strategy answering either r_i^1 or r_i^2 must have length

$$|b_i^*| \leq 1 - \frac{\Delta_i}{2} = (1 + \Delta_i) - \frac{3}{2}\Delta_i \leq \frac{3}{2} - \frac{3}{2}\Delta_i \leq \frac{3}{2}|b_i|,$$

where we use the fact that $\Delta_i \leq 1/2$ and, thus, $1 + \Delta_i \leq 3/2$.

So far, we have assigned every $b_i \in B$ to some $b_i^* \in B^*$ answering one of b_i's flanking requests, such that $|b_i^*| \leq (3/2)|b_i|$. Let now A_i refer to the set of all broadcasts assigned to b_i^*, $E(A_i)$ to energy consumption due to sending them. Consider a single broadcast b_i^* sent by the offline strategy and let $r_1, \ldots, r_k$ be the requests answered by it, ordered by their release times. Let t be the release time of r_1, d the deadline of r_k. Each $b \in A_i$ that is assigned to b_i^* must be flanked by one of $r_1, \ldots, r_k$ and, thus, must have starting time between t and d.

If $|b_i^*| > 1/2$, we know that $d - t \leq 2 - |b_i^*| < 3/2$. Additionally, $|b| \geq 1/2$ holds for every $b \in B$, which yields that $|A_i| \leq 3$. Thus, we obtain

$$E(A_i) = \sum_{b \in A_i} |b|^{-\alpha+1} \leq 3\left(\frac{2}{3}|b_i^*|\right)^{-\alpha+1} = 2 \cdot \left(\frac{3}{2}\right)^{\alpha} |b_i^*|^{-\alpha+1}.$$

If $|b_i^*| \leq 1/2$, we only have that $d - t \leq 2 - |b_i^*| < 2$ and, thus, $|A_i| \leq 4$. However, we also know that in

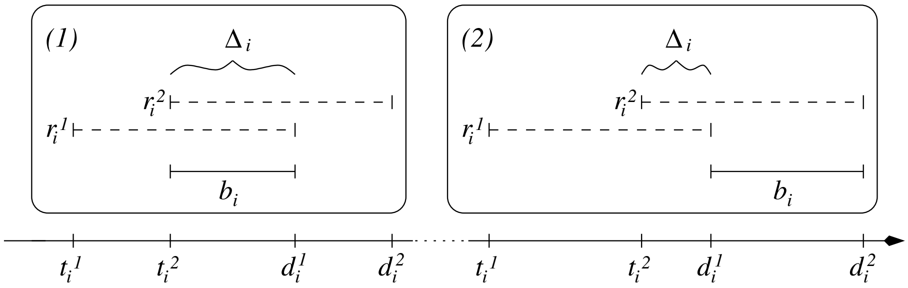


Figure 3: Broadcast b_i is flanked by requests r_i^1 and r_i^2.

this case $|b| \geq |b_i^*|$ for all $b \in A_i$ and may then write that

$$E(A_i) = \sum_{b \in A_i} |b|^{-\alpha+1} \leq 4 \cdot |b_i^*|^{-\alpha+1} \leq 2 \cdot \left(\frac{3}{2}\right)^{\alpha} |b_i^*|^{-\alpha+1},$$

where we use that $\alpha \geq 2$. Finally, let E_{ON} and E_{OPT} again refer to overall energy consumption on sequence R. Let $m = |B^*|$. Taking into account aborted broadcasts, we have

$$\begin{aligned} E_{ON} &\leq \frac{\alpha}{\alpha-1} \sum_{b \in B} |b|^{-\alpha+1} = \frac{\alpha}{\alpha-1} \sum_{i=1}^{m} \sum_{b \in A_i} |b|^{-\alpha+1} \\ &\leq \frac{\alpha}{\alpha-1} \sum_{i=1}^{m} 2 \cdot \left(\frac{3}{2}\right)^{\alpha} |b_i^*|^{-\alpha+1} = \frac{2\alpha}{\alpha-1} \left(\frac{3}{2}\right)^{\alpha} \cdot E_{OPT}, \end{aligned}$$

which proves the claim. □

Theorem 5. *The expected competitive ratio of every (randomized) online algorithm is* $\omega((2-\varepsilon)^{\alpha})$ *for any* $\varepsilon > 0$.

Proof: Theorem 4 shows that the claim holds for deterministic algorithms. Therefore, we will extend its proof to hold for any randomized algorithm A as well. Hence, we assume A to have a competitive ratio of $O((2-\varepsilon)^{\alpha})$ for an arbitrary constant $\varepsilon > 0$. The proof of the cited theorem uses Lemma 3 which relies on a case inspection in each step of the construction of the worst case instance depending on the gap's position. Obviously, as an oblivious adversary we do not know the random bits algorithm A uses when we have to fix the instance. Consequently, we cannot use this case inspection that easily. However, we will modify our approach to fit this situation as well.

First, we point out that underestimating the length of δ in the cited proof does not influence the proof's correctness: In this case the gap's relative length increases each step as well. However, the length of the series sufficient to show the contradiction increases accordingly. Consequently, we assume for the construction of the input instance that the relative length increases exactly by ε' each iteration. Request r_i will still be of length 2^{-i}. We start by issuing requests r_0 and r_1 in exactly the same way as we do in the cited proof. Let now $r_i = (t_i, d_i)$ and $(t_i + d_i)/2$ refer to the middle of request r_i for $i \geq 3$. With probability $1/2$ set $r_{i+1} = (m_i + (\delta+\varepsilon') \cdot |r_{i+1}|, d_i + (\delta+\varepsilon') \cdot |r_{i+1}|)$, otherwise set $(m_i - (\delta-\varepsilon') \cdot |r_{i+1}|, d_i - (\delta-\varepsilon') \cdot |r_{i+1}|)$. We call the extension of the instance by one additional request *a step*. The number of steps, i. e., the number of request contained in this instance, will be fixed in the course of this proof.

As the length of the last broadcast dominates the cost of the solution, we know that the expected length of the broadcast answering r_i must not fall below $(1/2 + \varepsilon') \cdot |r_i|$. Take a look at the gap's relative length and position in a single step. It is not difficult to see that for a single step our construction mimics the construction done in the proof of Theorem 4 with probability $1/2$. As the random choice in each step is done independently, we conclude that this holds simultaneously for the first ℓ steps with probability at least $(1/2)^{\ell}$. If we assume this to happen, we observe that the gap's relative length increases by ε' each step due to the linearity of the expected value. Therefore, substituting $\ell = 1/(2\varepsilon')$ gives rise to the conclusion that the expected relative length of the gap reaches $1/2$ in at least one of the first ℓ steps with probability at least $(1/2)^{\ell} = \Omega(1)$. Note, that both this probability and the number of steps is constant. Hence, there has to be a step in which the expected relative length of the gap reaches $1/2$ with probability at least $((1/2)^{\ell})/\ell$, which still is constant. We stop our construction right after this step. Now we observe that the expected length of the broadcast answering the last broadcast falls below $(1/2 + \varepsilon') \cdot |r_i|$ with constant probability. This is a contradiction to the assumed competitive ratio and, therefore, completes the proof of this theorem. □

Theorem 6. *Let E_{MM} denote the energy consumption of algorithm* ONLINE-MM *on any sequence of requests of varying length between ℓ and $c\ell$ for some fixed $c \geq 1$, E_{OPT} the value of an optimal offline solution on the same sequence. It holds that*

$$E_{MM} \leq (4c-1)^{\alpha} \cdot E_{OPT}.$$

Proof: Let $B_1, \ldots, B_n$ denote the sequences of broadcasts sent by algorithm ONLINE-MM on any given sequence of requests. We first observe that every request $r = (t, d, m)$ completely contains some sequence B_i including message m. To see this, note, that when request r arrives at time t, the algorithm sets $\tau = \min\{\tau, d - \ell/2\}$ (or ρ, respectively) and, thus, eventually there will be a sequence of broadcasts starting after time t, finishing before time d and containing m. We shall say that request r is assigned to group B_i. Assume now that each B_i consists of broadcasts for k_i different messages and let E_i refer to the corresponding energy consumption. It clearly holds that

$$E_i = k_i \cdot \left(\frac{2k_i}{\ell}\right)^{\alpha-1}.$$

By R_i we refer to the set of requests assigned to the respective group of broadcasts B_i. If the first broadcast in group B_i starts at time t_i, then requests in R_i are posed between times $t_i - (c-1/2)\ell$ and t_i and, thus, are contained in the interval $\mathcal{I}_i = [t_i - (c-1/2)\ell, t_i + c\ell]$. We note that $t_{i+1} \geq t_i + \ell/2$.

Let us now split up the groups B_i and define $r = \lceil 4c-2 \rceil$ and

$$\mathcal{B}_k = \{B_k, B_{k+r}, B_{k+2r}, \ldots\}$$

for $k = 1, \ldots, r$. Let $\mathcal{B}_j$ be the most expensive set of groups sent by algorithm ONLINE-MM among these. The overall energy consumption E_{MM} of our algorithm is then bounded by

$$E_{MM} \leq r \sum_{B_i \in \mathcal{B}_j} k_i \cdot \left(\frac{2k_i}{\ell}\right)^{\alpha-1}.$$

On the other hand, for any B_i, $B_k \in \mathcal{B}_j$ with $i < k$ it holds that $t_k - (c-1/2)\ell \geq t_i + (c-1/2)\ell$ and, thus, the corresponding intervals $\mathcal{I}_i$ and $\mathcal{I}_k$ do not intersect. Let E_{OPT} denote the energy consumption of an optimal offline strategy on the given input. It is straightforward to argue that E_{OPT} is lower bounded by the energy needed to service requests linked to groups in $\mathcal{B}_j$ only and that inside each interval $\mathcal{I}_i$ at least k_i broadcasts need to be sent. Hence, we can write that

$$E_{OPT} \geq \sum_{B_i \in \mathcal{B}_j} k_i \cdot \left(\frac{k_i}{(2c-1/2)\ell}\right)^{\alpha-1}.$$

This immediately yields that

$$E_{MM} \leq (4c-1)^\alpha \cdot E_{OPT},$$

which finishes the proof. $\square$

A.1 Proof of Theorem 7

Theorem 7 *The competitive ratio of any (randomized) online algorithm capable of speed-up is at least* $\omega((\gamma-\varepsilon)^\alpha)$ *for* $\gamma = (5+5\sqrt{5})/(5+3\sqrt{5}) > 1.38$ *and any* $\varepsilon > 0$. *If all requests are of identical length the competitive ratio is at least* $\Omega(1.09^\alpha)$.

For the sake of clarity, we split up the proof of this theorem into two parts. In the first one we prove the lower bound of $\omega((\gamma-\varepsilon)^\alpha)$ for any $\varepsilon > 0$, while the bound of $\Omega(1.09^\alpha)$ will be shown in the second part.

A.1.1 Lower bound $\omega((\gamma-\varepsilon)^\alpha)$ for arbitrary requests

We will start by showing this bound for deterministic algorithms. Similarly to the proof of Theorem 4 we assume there is an algorithm A that has a competitive ratio of at most $O((\gamma-\varepsilon)^\alpha)$ for a constant $\varepsilon > 0$ and construct a sequence $R_0, R_1, \ldots$ of input instances that leads to a contradiction. Again, instance R_j consists of requests $r_0, r_1, \ldots, r_j$. Let $\phi := (1+\sqrt{5})/2$ denote the golden section. We issue request r_0 with length 1 at time 0. Generally, request r_{i+1} is issued with length $\psi^{i+1} := ((3-\sqrt{5})/(\sqrt{5}-1))^{i+1}$ when ϕ^{-1} parts of request r_i have passed. We already argued in the cited proof that the behavior of algorithm A is identical on instance R_j and on R_{j+1} up to the release time of request r_{j+1} as the algorithm is online.

Next, we will limit the energy of an optimal schedule for instance R_n from above. We answer the last request with a broadcasts that covers the whole request. This induces cost of $\psi^{-(\alpha-1)n}$. As each request r_{i+1} is issued after ϕ^{-1} parts of request r_i have passed, we can answer each remaining request r_i with a broadcast that has length $\phi^{-1} \cdot \psi^i$. Therefore, the cost Opt_n of an optimal schedule for instance R_n sums up to at most

$$\begin{aligned}
\mathrm{Opt}_n &\leq \psi^{-(\alpha-1)n} + \sum_{i=0}^{n-1} \left(\phi \cdot \psi^{-i}\right)^{\alpha-1} \\
&= \left(\left(\frac{\sqrt{5}-1}{3-\sqrt{5}}\right)^{\alpha-1}\right)^n + \frac{1+\sqrt{5}}{2} \cdot \frac{\left(\left(\frac{\sqrt{5}-1}{3-\sqrt{5}}\right)^{\alpha-1}\right)^n - 1}{\left(\frac{\sqrt{5}-1}{3-\sqrt{5}}\right)^{\alpha-1} - 1} \\
&\leq \left(\left(\frac{\sqrt{5}-1}{3-\sqrt{5}}\right)^{\alpha-1}\right)^n \cdot \left(2 + \frac{\left(\frac{1-\sqrt{5}}{2}\right)^{\alpha-1}}{\left(\frac{\sqrt{5}-1}{3-\sqrt{5}}\right)^{\alpha-1}}\right) \\
&= 3 \cdot \left(\left(\frac{\sqrt{5}-1}{3-\sqrt{5}}\right)^{\alpha-1}\right)^n = 3 \cdot \psi^{-(\alpha-1)n}.
\end{aligned}$$

Hence, the cost of an optimal schedule is dominated by the cost of the very last request. Consequently, in order to obtain a competitive ratio of $\Omega((\gamma-\varepsilon)^\alpha)$ the length of each broadcast of algorithm A on instance R_n must not be be below $(\gamma-\varepsilon)^{-1} \cdot \psi^n = (\gamma^{-1}+\varepsilon') \cdot \psi^n = (\phi^{-1}+\varepsilon'') \cdot \psi^n$ for some properly chosen constants $\varepsilon', \varepsilon'' > 0$. In particular, this holds for the broadcast answering request r_n.

Let us have a look at request r_i. At its beginning there are $\delta \cdot \psi^i$ time units not used to answer this broadcast for some $\delta \geq 0$. Remember, that this period of time is called a *gap* and δ denotes its relative length. Note, that $\delta = 0$ is possible. Due to our argumentation, the broadcast that answers r_i has to be of length at least $(\phi^{-1} + \varepsilon'') \cdot \psi^i$. Therefore, this theorem would be proved if we showed that δ exceeds $1 - \phi^{-1} - \varepsilon''$ on instance R_n for a finite n. To this end, we will show that the relative length δ of the gap increases with every newly added request. Figure 4 depicts this situation. When the algorithm becomes aware of request r_{i+1} at its release time it essentially has two choices of action: either it aborts the currently performed broadcast or it continues this broadcast until it is submitted completely. If the algorithm decides to abort the broadcast the new broadcast has to be finished at the deadline of request r_i. As there are at most $(1 - \phi^{-1}) \cdot \psi^i$ time units of request r_i left, this decision results in a competitive ratio of

$$\frac{1}{3} \cdot \left(\frac{\psi^{-(i)}}{\psi^{-(i+1)}} \right)^{\alpha - 1} = \frac{1}{3} \cdot \left(\frac{\frac{3-\sqrt{5}}{\sqrt{5}-1}}{1 - \frac{2}{1+\sqrt{5}}} \right)^{\alpha - 1} = \frac{1}{3} \cdot \left(\frac{1 + \sqrt{5}}{2} \right)^{\alpha - 1} = \Omega(\phi^\alpha) = \omega(\gamma^\alpha).$$

Hence, aborting the currently performed broadcast b that answers request r_i is no option and we henceforth assume that algorithm A does not abort it when r_{i+1} is issued. Nevertheless, it can change the speed at which b is submitted. As we want to show that the gap's relative length increases, decreasing the broadcast speed is in our interest. Therefore, we assume that the algorithm increases the speed. We will delimit this speed from above to show that the relative length of the gap of request r_{i+1} has increased at least by a constant amount—compared to the relative length of the gap of request r_i.

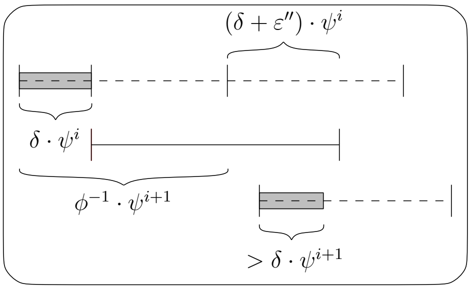


Figure 4: The relative length δ of the gap increases when adding request r_{i+1}.

As broadcast b cannot be finished before $\phi^{-1} + \delta + \varepsilon''$ parts of request r_i have passed, at least $(\delta + \varepsilon'')/(\phi^{-1} + \delta + \varepsilon'')$ parts of broadcast b remain for submission when request r_{i+1} is issued. In order to submit them in time t a speed of $(\delta + \varepsilon'')/((\phi^{-1} + \delta + \varepsilon'') \cdot t)$ is necessary that yields an energy consumption of

$$E(t) := \left(\frac{1}{t} \right)^{\alpha - 1} \cdot \left(\frac{\delta + \varepsilon''}{\phi^{-1} + \delta + \varepsilon''} \right)^{\alpha - 1}.$$

Even when pessimistically ignoring the energy consumed up to that point energy $E(t)$ must not exceed the energy consumed by request r_{i+1}, i. e., $\psi^{-(i+1)(\alpha-1)}$, by more than a factor $(1/(\phi^{-1} + \varepsilon''))^{\alpha-1}$ in order to stick to the claimed competitive ratio. Doing some algebra reveals

$$\left(\frac{1}{t} \right)^{\alpha - 1} \cdot \left(\frac{\delta + \varepsilon''}{\phi^{-1} + \delta + \varepsilon''} \right)^{\alpha} < \left(\frac{1}{\phi^{-1} + \varepsilon''} \cdot \frac{1}{\psi^{i+1}} \right)^{\alpha - 1}$$

$$\Rightarrow \qquad t > \sqrt[\alpha-1]{\frac{\delta + \varepsilon''}{\phi^{-1} + \delta + \varepsilon''}} \cdot \psi^{i+1} \cdot \frac{(\delta + \varepsilon'')(\phi^{-1} + \varepsilon'')}{\phi^{-1} + \delta + \varepsilon''}.$$

By increasing α the root in the inequality above converges to 1 and we see that the gap increases with each newly issued request if $(\delta + \varepsilon'')(\phi^{-1} + \varepsilon'')/(\phi^{-1} + \delta + \varepsilon'') > \delta$ holds for all δ. This is equivalent to $(\varepsilon'')^2 + \varepsilon''\phi^{-1} > \delta^2$. As soon as δ exceeds $1 - \phi^{-1} - \varepsilon''$ by a constant amount the theorem is shown. Therefore, we assume this not to happen and transform inequality above into $(\varepsilon'')^2 + \varepsilon''\phi^{-1} > (1 - \phi^{-1} - \varepsilon'')^2$. Finally, this gives rise to

$$\varepsilon'' > \frac{1 + \phi^{-1^2} - 2\phi^{-1}}{2 - \phi^{-1}} = \frac{3\sqrt{5} - 5}{5\sqrt{5} + 5} = \gamma^{-1} - \phi^{-1}.$$

Since $\varepsilon'' = \gamma^{-1} - \phi^{-1} + \varepsilon'$ holds, this inequality is fulfilled for each $\varepsilon' > 0$. Therefore, the gap's relative length increases each step by a constant amount and, consequently, exceeds $1 - \phi^{-1} - \varepsilon''$ after a finite number of steps. Hence, a competitive ratio of less than $\Omega(\gamma^\alpha)$ leads to a contradiction and the lower bound is proven for deterministic algorithms.

Therefore, we merely need to extend its proof to hold for randomized algorithms as well. To this end we assume that there is an algorithm A that achieves a competitive ratio of $O((\gamma - \varepsilon)^\alpha)$ for a constant $\varepsilon > 0$ and use exactly the same input instances as in the deterministic version. The algorithm has to use at least $(\gamma^{-1} + \varepsilon')$ parts of the last request with probability at least $1 - \gamma \cdot (\gamma^{-1} + \varepsilon')^\alpha$ as otherwise it could not achieve the claimed competitive ratio. As this holds for each instance R_i it holds with probability at least $(1 - \gamma \cdot (\gamma^{-1} + \varepsilon')^\alpha)^{n+1}$ for each request $r_0, r_1, \ldots, r_n$ of instance R_n. As the length n of the series that suffices to obtain a contradiction depends only on ε and ε' but not α, this probability converges to 1 as α increases. Hence, the claims follows directly from the law of total probability.

A.1.2 Lower bound $\Omega(1.09^\alpha)$ for requests of identical length

It remains to show the claimed lower bound of 1.09^α for randomized algorithms. As randomized algorithms tend to be "hard to control", we will use Yao's minimax principle [13] for the sake of bounding the competitive ratio of any online algorithm from below. This theorem allows us to look at deterministic online algorithm only, so that we do not have to bother with randomization at all. However, in order to show a lower bound we have to fix an input distribution $\mathcal{R}$, compute the expected cost E_{OPT} it causes when presented to an optimal offline algorithm and, finally, bound the cost E_{ON} of an optimal deterministic online algorithm A from below that is aware of $\mathcal{R}$. The last part is the most difficult one. Then, the ratio E_{ON}/E_{OPT} is a lower bound for the competitive ratio of any randomized online algorithm. We point out that th deterministic algorithm A is aware of $\mathcal{R}$ but does not know the actual choice of the random bits.

First of all, we have to specify the input distribution $\mathcal{R}$. The key idea to show the lower bound is to "force" the algorithm A to make a mistake before it knows which input instance it was given. Hence, we define the following two input instances. Instance R_1 consists of one request at time 0 with deadline 1. For a given $t \in [0, 1]$—that we will specify later on—we obtain R_2 from R_1 by adding a request with release time t and deadline $1 + t$. Observe that both requests have the length 1. Next, we specify the input distribution $\mathcal{R}$ by saying that instance R_1 is chosen with probability $1 - p$ and instance R_2 otherwise. We will fix p appropriately later on. As we are about to compare the expected cost induced by an online algorithm to the expected cost induced by an optimal algorithm for $\mathcal{R}$, we will compute the cost of an optimal schedule for $\mathcal{R}$ in the following lemma.

Lemma 4 *The expected cost E_{OPT} of an optimal broadcast schedule on $\mathcal{R}$ sum up to*

$$E_{OPT} = \begin{cases} (1-p) + 2 \cdot p \cdot \left(\frac{2}{1+t}\right)^{\alpha-1} & \text{if } t \geq \frac{\sqrt[\alpha-1]{2}-1}{\sqrt[\alpha-1]{2}+1} \\ (1-p) + p \cdot \left(\frac{1}{1-t}\right)^{\alpha-1} & \text{otherwise.} \end{cases}$$

In the first case the optimal solution consists of two broadcasts $[0, (1+t)/2]$ and $[(1+t)/2, 1+t]$. In the latter case there is just one broadcast $[t, 1]$.

Proof: Obviously, the single request of instance R_1 can be answered by a broadcast $[0,1]$ with cost 1. Since this instance is chosen with probability $1-p$ it yields a addend $1-p$ to the expected cost.

There are two possibilities to answer the two requests of instance R_2: Either by one broadcast that answers both requests simultaneously, or by one broadcast for each request. Due to the given release times and deadlines the optimal broadcast that answers both is $[t,1]$. Therefore, its cost is $(1/(1-t))^{\alpha-1}$. On the other hand, as a consequence of Lemma 1 it is not hard to see that the two broadcast $[0,(1+t)/2]$ and $[(1+t)/2,1]$ consume the least energy among all schedules that answer the requests by using two broadcasts within $[0,1+t]$. In this case, the cost sums up to $(2/(1+t))^{\alpha-1}$. Hence, we see that two broadcasts are cheaper if and only if $t \geq (\sqrt[\alpha-1]{2}-1)/(\sqrt[\alpha-1]{2}+1)$ holds. Either way, the cost induced by R_2 is weighted with p in the expected cost of $\mathcal{R}$. This shows the lemma. $\square$

Now we can complete the proof of Theorem 7. As we are intending to use Yao's minimax principle it is time to specify the parameters of $\mathcal{R}$ entirely. We use the above defined probability distribution on the instances R_1 and R_2 and use the parameters $t := \sqrt{2}-1$ and $p := ((1+t)/2)^{\alpha-1}$.

According to Lemma 4, for all $\alpha \geq 2$ an optimal broadcast schedule consists of two (identically sized) broadcasts and yields cost of

$$1-\left(\frac{1+t}{2}\right)^{\alpha-1}+2\cdot\left(\frac{1+t}{2}\right)^{\alpha-1}\cdot\left(\frac{2}{1+t}\right)^{\alpha-1}<3.$$

Let us apply Yao's minimax principle and have a closer look at it. The deterministic online algorithm A used in it does know the input distribution $\mathcal{R}$. In particular it knows the value of t. However, it is unaware of the choice of the random bits, i. e., the choice between R_1 and R_2. This holds up to time t as it can distinguish these two instances by then. Due to the convexity of the cost function the optimal algorithm does change its transmission speed only at time t. Hence, we can assume without loss of generality that the algorithm A is acting according to the scheme shown in Figure 5.

1. Start broadcast $[0,t']$ and perform it up to time t.
2. If there is a new request at time t, reevaluate the situation and choose the more favorable of the following two strategies:
   - a Continue the current broadcast changing its speed accordingly such that it ends at time t''. Subsequently, perform another broadcast until time $1+t$.
   - b Abort the current broadcast and start a new one, that answers both requests simultaneously within their deadlines.
3. If there is no new request at time t, change the speed of the current broadcast such that it finishes at time 1.

Figure 5: Scheme of an optimal algorithm for input distribution $\mathcal{R}$.

Choosing $t' < t$ does not result in an optimal algorithm since the interval $[t',t]$ is not used in instance R_1 as well as in R_2 although using it is not prohibited by any deadline. Accordingly, we obtain a better solution by redefining $t'_{\text{new}} := (t+t')/2$. Hence, we assume $t' \geq t$ in the following.

We will adjust the parameters t' and t'' such that the expected cost is minimal and bound this cost from below. In this way, we obtain the optimal online algorithm for $\mathcal{R}$ as well as a lower bound on its cost. For this purpose, we first focus on finding the optimal t'' for given t'.

As the choice of t'' is only affecting Step 2a, we assume for the moment that A chooses this kind of schedule. Thus, instance R_2 was presented and answered by two broadcasts. Up to time t a portion of

$1 - t/t'$ of the first request has already been answered. Hence, the optimal speed for the interval $[t, t'']$ is exactly $(1 - t/t')/(t'' - t)$. Consequently, the function g defined by

$$g_{t,t'}(t'') := t \cdot \left(\frac{1}{t'}\right)^{\alpha} + (t'' - t) \cdot \left(\frac{1 - \frac{t}{t'}}{t'' - t}\right)^{\alpha} + \left(\frac{1}{1 + t - t''}\right)^{\alpha - 1}$$

represents the cost induced by the two broadcasts. It is straightforward to verify that this is minimal for $t_0'' := ((2t + 1)t' - t - t^2)/(2t' - t)$ and yields cost of $g_{t,t''}(t_0'') = (t + (2t' - t))/(t')^{\alpha}$. Now, that we know the optimal choice for t'' we can focus on finding the optimal t'. Step 2b results in cost $t \cdot \left(\frac{1}{t'}\right)^{\alpha} + \left(\frac{1}{1-t}\right)^{\alpha-1}$ and Step 3 in cost $t \cdot \left(\frac{1}{t'}\right)^{\alpha} + \left(\frac{1-\frac{t}{t'}}{1-t}\right)^{\alpha} \cdot (1-t)$. Since the condition of Step 2 is fulfilled with probability $1-p$ and the one of Step 3 with probability p, we have gathered all information needed to make up a connection between t' and the expected cost. Therefore, the cost induced by the choice of time t' is denoted by the function c that is defined by

$$\begin{aligned} c(t') &:= \operatorname{cost}(\text{Step 1}) + p \cdot \min\{\operatorname{cost}(\text{Step 2a}), \operatorname{cost}(\text{Step2 b})\} \\ &= t \cdot \left(\frac{1}{t'}\right)^{\alpha} + (1 - p)\left(\frac{1 - \frac{t}{t'}}{1 - t}\right)^{\alpha} \cdot (1 - t) + p \cdot \min\left\{\left(2 - \frac{t}{t'}\right)^{\alpha}, \left(\frac{1}{1 - t}\right)^{\alpha - 1}\right\}. \end{aligned}$$

Evaluating the min-term in the definition of this function might prove a little bit tricky. However, let us recall that we are interested in finding a lower bound on the competitive ratio of an arbitrary online algorithm. Hence, a lower bound on the minimal value of f will suffice. It holds that

$$\left(2 - \frac{t}{t'}\right)^{\alpha} < \left(\frac{1}{1 - t}\right)^{\alpha - 1} \Leftrightarrow t' < \frac{t}{2 - (1 - t)^{\frac{1-\alpha}{\alpha}}} =: c_{\alpha}.$$

To obtain a lower bound on this term we define the two helping functions c_1 and c_2 by

$$\begin{aligned} c_1(t') &:= t \cdot \left(\frac{1}{t'}\right)^{\alpha} + (1 - p) \cdot \left(\frac{1 - \frac{t}{t'}}{1 - t}\right)^{\alpha} \cdot (1 - t) \\ \text{and } c_2(t') &:= t \cdot \left(\frac{1}{t'}\right)^{\alpha} + p\left(2 - \frac{t}{t'}\right)^{\alpha}. \end{aligned}$$

As all addends in c are positive—we already reasoned that t' is bounded below by t—we obtain:

$$c(t') \geq \begin{cases} c_1(t') & \text{if } t' \geq c_{\alpha} \\ c_2(t') & \text{otherwise.} \end{cases}$$

It is not difficult to verify that these functions c_1 and c_2 take their minimal value at $(1 + (\sqrt[\alpha-1]{1 - p} - 1) \cdot t)/(\sqrt[\alpha-1]{1 - p})$ and $\frac{1}{2} \cdot (t + \sqrt[\alpha-1]{\alpha/(p \cdot (\alpha - 1))})$, respectively. Using $p = ((1 + t)/2)^{\alpha-1}$ and doing some algebra suffices to verify that the cost $c(t'_{\min})$ of the algorithm A is bounded below by $\Omega(1.09^{\alpha})$ holds. We conclude that an application of Yao's Theorem finishes the proof of this theorem as Lemma 4 states that the expected optimal cost on $\mathcal{R}$ is constant.