On Multi-Channel Data Broadcast Scheduling

Aaron T. Hawkins and Weizhen Mao
Department of Computer Science, College of William & Mary, Williamsburg, VA 23187-8795, USA
{*ahawkins,wm*}*@cs.wm.edu*

Abstract

We consider the following communication problem. A single server is given access to a set of data items, channels over which items may be broadcast, and requests that arrive at various times for the items. At each system tick, the server must select an item to broadcast for each available channel, thus satisfying all requests for that item, with the goal of minimizing the total wait time of all requests. Though several on-line scheduling algorithms have been developed for this problem, these algorithms have been studied primarily through simulation experiments with little emphasis on theoretical analysis. In this paper we prove a general lower bound to the competitive ratios of all on-line algorithms for the problem in the context of unit-sized items and multiple channels. We also provide competitive analysis of two well-known on-line algorithms, First Come First Served and Most Requested First. The competitive ratios obtained for the two algorithms are consistent with previous simulation experiments. The fact that the competitive ratios match our general lower bound indicates the optimality of the algorithms.

1 Introduction

Increasingly high user demands on data dissemination applications have motivated technology improvements in the areas of mobile computing, satellite broadcasting, and cable networks. While these systems provide high bandwidth, their infrastructures are asymmetric in that the amount of information that may be transferred is much larger from server to client than vice versa. Such systems create environments in which traditional unicasting techniques are inapplicable or not scalable [1, 2]. In contrast, data broadcasting is an efficient method of providing large-scale data delivery by simultaneously satisfying requests from multiple clients. Several commercial entities use the data broadcasting method, including the Hughes DirectPC System, the Intel Intercast System, the Hybrid System, and the AirMedia System [2].

In this paper we consider a data broadcasting system in which clients send requests for data items (pages) to a single server. Such a system is referred to as pull-based because the transfer of information is initiated by a client as opposed to push-based delivery in which information is sent without any specific request [1]. In a pull-based system, once a channel becomes available, the server must select a page to send based on the requests that have arrived. The selected page is sent on the idle channel to the entire client population, satisfying all requests for that page. We consider the scheduling problem that arises in this situation.

2 Model Definition

We formally define the general problem of data broadcast scheduling as follows. We are given a single server with access to a set of distinct pages. Requests for those pages arrive to the server at various times and are placed into the server's queue. Assume that the number of requests is n. The server has $c \geq 1$ channels over which pages may be broadcast to all clients simultaneously. A tick at time t is an interval $[t, t+1)$ defined as the amount of time required to broadcast an item of unit size over a single channel. Without loss of generality we assume that the system clock begins at time $t = 0$.

Because the server wishes to minimize the total wait time, which is also the amount of time spent by all requests in the queue, the determination of which page to send each tick is a scheduling decision. Furthermore, scheduling algorithms used to make these decisions can only do so based upon the requests currently in the queue with no knowledge about what requests may arrive later. As such, these on-line algorithms are necessarily based on local information and will provide approximate solutions [5].

Suppose the single server has access to m distinct pages of various lengths. Let l_j be an integer-valued length of page j such that if $l_j = 1$, then j is a unit-length page that can be completely sent in a single system tick. A length of 2 will require 2 ticks and so on. A channel may only be used to send a single page and, once assigned, cannot be preempted to send a different page until the current one is completely sent. Thus, once a page j is assigned to a channel k, that channel will not be available until l_j system ticks later. The page broadcast at time t on channel k is denoted b_{tk}. It is therefore possible that as many as c and as few as 0 pages will begin broadcasting each tick. When $m \leq c$, the problem is

trivial. When $m > c$, however, the server will have to select pages to send each tick based upon the requests that have arrived and its scheduling algorithm.

A request i is described by a_i, the arrival time of the request, and p_i, the requested page. It is assumed that the server is ignorant of any arrival rate distribution or the distribution governing what page will be requested by an arrival. We say that request i is satisfied at time s_i if s_i is the smallest integer greater than a_i such that $b_{s_i k} = p_i$ for some channel k. Thus the wait time of request i is defined as $w_i = s_i - a_i$. Note that $w_i \geq 1$ for any request in any schedule. Given that the server must process n requests, the sum of the wait times for any schedule will be $\sum_{i=1}^{n} w_i \geq n$.

For the remainder of the paper, we focus our attention on the performance of two algorithms, First Come First Served (FCFS) and Most Requested First (MRF), in the context of c channels and unit-sized pages. Under these parameters, the single server will select as many as c pages to broadcast each system tick. Because all the pages are unit-sized, every channel will again be available to send another page the next tick. We note that although these two algorithms were originally defined for the single channel problem, no modifications to the algorithms are necessary to utilize them in the presence of c channels. Though all selected pages will be broadcast simultaneously, we may view the page selection process as c sequential applications of the same algorithm used in the single channel case.

3 Competitive Analysis

A push-based server must anticipate the needs of its clients based on item frequencies or other stochastic models [3]. As mentioned above, however, the on-line algorithms used to make pull-based scheduling decisions have access to no information beyond those requests that have already arrived to the queue. These algorithms are contrasted with off-line algorithms which are aware of the entire sequence of requests in advance. That is, on-line algorithms know only the present, but their off-line counterparts know the future [4]. It is intuitive, then, that the on-line algorithms are at a disadvantage and will, in general, not perform as well as off-line algorithms. In addition, it is somewhat more difficult to identify the optimality of an on-line algorithm since, from a worst-case perspective, it is always possible to construct sequences of requests that can ruin the performance of the on-line algorithm. The optimal off-line algorithm, however, will be easily identified by merit of being the algorithm which minimizes the total wait time of a schedule for each sequence of requests. In the absence of any assumptions restricting the stochastic behavior of the request sequence, we choose a worst-case analysis approach called competitive analysis.

An on-line algorithm A is said to be r-competitive if for any sequence of requests σ, $cost_A(\sigma) \leq r \cdot cost_{OPT}(\sigma)$, where *cost* refers to the value of the objective function in the solution produced by the on-line algorithms A or the optimal off-line algorithm OPT. This r value is known as the competitive ratio. Competitive analysis does not measure the time efficiency of an on-line algorithm, but rather the quality of solutions produced by such an algorithm when compared to the optimal solution on the same input sequence. For the competitive ratio to be meaningful, it is necessary to bound the value of r from above and below. Of these two bounds, the lower bound is typically simpler to provide in that it requires only a single instance in which algorithm A produces a solution that is at least r times the solution provided by OPT. Establishing an upper bound requires proving that the competitive ratio is no larger than r for all σ. When the lower bound and the upper bound match, the competitive ratio is said to be tight.

Though the model definition is more general, the proof sketches in the next two sections focus on the performance of two algorithms, FCFS and MRF, in the context of c channels and unit-sized pages. Typically, each proof in its entirety will involve two cases: a case in which the number of distinct pages m is less than or equal to the number of channels c and a case in which m is greater than c. The first case is trivial in all proofs and is not described in the proof sketches. However, the ratios derived do include the term $d = \min(c, m)$, reflecting the possibility of both cases.

4 First Come First Served

In FCFS, the page requested by the earliest arriving (oldest) request in the queue is chosen by the server every time a channel becomes idle. For any instance, let $\sum_{i=1}^{n} w_i(FCFS)$ and $\sum_{i=1}^{n} w_i(OPT)$ be the total wait times of the FCFS and optimal schedules, respectively.

Theorem 1. $\sum_{i=1}^{n} w_i(FCFS) \leq \frac{m}{d} \sum_{i=1}^{n} w_i(OPT)$.

We prove the upper bound by contradiction. Assume that there exists some request with a wait time larger than $\frac{m}{d}$. This assumption would cause the FCFS algorithm to broadcast some page twice before some other page in the queue has been broadcast at all, thus contradicting the behavior of FCFS.

Theorem 2. *There exists at least one instance for which* $\sum_{i=1}^{n} w_i(FCFS) = \frac{m}{d} \sum_{i=1}^{n} w_i(OPT)$.

We prove the lower bound by establishing that there is at least one instance for which the total wait

time of the FCFS schedule is equal to $\frac{m}{d}$ times that of the OPT schedule. We construct this instance as follows. The first m requests arrive at time $t = 0$, each for a different page. Knowing the behavior of FCFS, we further design the arrival sequence so that there is exactly one request per page waiting at any time t. This instance causes the FCFS algorithm to select pages to broadcast in a circular order whereas the optimal schedule will select a page to broadcast such that it will satisfy as many requests as possible. The first m requests will be satisfied with the same amount of cumulative wait time in each schedule, but the remaining $n-m$ requests will experience more wait time under the FCFS schedule. We can show that this additional wait time creates a ratio of $\frac{m}{d}$ between the FCFS schedule and the optimal schedule.

5 Most Requested First

In MRF, the page requested by the largest number of unsatisfied requests in the queue is selected by the server every time a channel becomes idle. For any instance, let $\sum_{i=1}^{n} w_i(MRF)$ and $\sum_{i=1}^{n} w_i(OPT)$ be the total wait times of the MRF and optimal schedules, respectively.

Theorem 3. $\sum_{i=1}^{n} w_i(MRF) \leq \frac{m}{d} \sum_{i=1}^{n} w_i(OPT)$.

We prove the upper bound by comparing the number of pages satisfied by each page broadcast at any tick to those left unsatisfied. This comparison is effective because the total wait time of the schedule can be alternatively viewed as the sum of all requests left unsatisfied at the end of each system tick. Due to the behavior of MRF, the number of requests satisfied by broadcasting a page is at least as large as the number of unsatisfied requests dependent upon any other page not selected for broadcast. Thus the total wait time of the schedule can be described as the summation of wait times for each system tick where each tick's wait time is the number of requests still waiting at that tick added to the number of requests that arrived. It can be shown that this summation is at most $\frac{m}{d}$ times that in the optimal schedule.

Theorem 4. *There exists at least one instance for which* $\sum_{i=1}^{n} w_i(MRF) = \frac{m}{d} \sum_{i=1}^{n} (OPT)$.

We prove the lower bound by using the same instance and strategy illustrated in the lower bound proof for FCFS. Note that this is made possible because the constructed instance results in exactly one request per page waiting at any time t. Since MRF will view all these requests as having the same priority, we can assume without loss of generality that MRF will choose to broadcast pages in the same order selected by FCFS.

6 General Lower Bound

In this section, we consider the general lower bound for multi-channel data broadcast scheduling.

Theorem 5. *For any on-line algorithm for page selection with c channels and unit-sized pages, its competitive ratio is at least $\frac{m}{d}$, where m is the number of pages and $d = min(c, m)$.*

We use an adversary argument and assume that the input instance is provided by the adversary. The adversary is a theoretical agent that uses its understanding of the behavior of an algorithm to choose inputs that force the worst-case performance of that algorithm. Because the algorithms under study are on-line, they have no knowledge of the request sequence except for the requests that have already arrived by the time a scheduling decision is to be made. Thus, the adversary is able to wait until the algorithm has made its scheduling decisions each broadcast tick before deciding which pages to request for the upcoming tick. This enables the adversary to construct a sequence of requests that will produce the highest possible cost from an arbitrary algorithm A while at the same time, the optimal algorithm OPT produces a small cost. Since algorithm A is arbitrary throughout the proof, such a lower bound holds for all on-line algorithms for the problem.

The adversary initially makes m requests with arrival times of $a_i = 0$ and requested page of $p_i = i$ for $i = 1, 2, \ldots, m$. The on-line algorithm will select $d = \min(c, m)$ pages to broadcast each tick, and on the first tick this will be pages $b_{11}, b_{12}, \ldots, b_{1d}$. For each successive broadcast tick the adversary will generate d requests, a request for each page that was just broadcast. Specifically, requests $m + d(t-1) + 1, m + d(t-1) + 2, \ldots, m + d(t-1) + d$ arrive at time t requesting pages $b_{t1}, b_{t2}, \ldots, b_{td}$ respectively, for $t = 1, 2, \ldots, \frac{n-m}{d}$. This creates a schedule in which for each page there is exactly one request waiting at any time t, for $1 < t \leq \frac{n-m}{d}$. After time $t = \frac{n-m}{d}$ no more requests will arrive and $m - d$ requests will remain to be satisfied. These remaining requests will be satisfied in $\lceil \frac{m-d}{d} \rceil$ additional ticks. That is, the schedule will be completed at the end of time $t = \frac{n-m}{d} + \lceil \frac{m-d}{d} \rceil$. This description provides enough information for the total wait time of all requests to be computed.

For the same input instance there exists an optimal schedule in which $w_i = 1$ for most requests. In the trivial case of $c \geq m$, the optimal schedule is equivalent to the on-line schedule. If $c < m$ then we must closely observe the input provided by the adversary. Because the adversary will always generate requests for a page to replace requests that have just been satisfied by the on-line algorithm, we note that

the on-line algorithm is never able to satisfy more than one request per page broadcast. In contrast, there exists a schedule constructed with knowledge of the future in which a page is selected to be broadcast at the tick in which most of the most immediate requests for that page have already arrived.

It is with this motivation that the optimal schedule is constructed. An algorithm with complete knowledge of the input generated by the adversary will choose to select such a page to broadcast that the difference between the current time and the arrival time of the next request for the same page is maximized. (The distance is assumed to be infinity if a request is never followed by a request for the same page.) In this way, requests generated by the adversary will tend to be for the same pages as other requests already arrived thus decreasing the number of unique pages requested.

At time $t = 0$ OPT knows that m unique requests have arrived at $t = 0$ and d additional requests will arrive at $t = 1$. Based on the above selection strategy OPT will choose d pages to broadcast at time $t = 1$. If any of these pages correspond to those pages to be requested at time $t = 1$, then necessarily it must be true that $m < 2d$. In this case it is obvious that all requests arriving before or at $t = 2$ will be satisfied at the end of $t = 2$, otherwise each page broadcast will not be requested at $t = 1$ and the number of unique requests at $t = 2$ will be $m - d$. That is, in contrast to the on-line schedule in which there was a request for every page throughout the schedule, the optimal schedule has already eliminated up to d unique requests each tick. By continuing this method of page selection, the optimal schedule is able to quickly satisfy all the initial requests that have arrived at or before time $\lceil \frac{m}{d} \rceil$.

The total wait time for the optimal schedule can also be computed. Given $\sum_{i=1}^{n} w_i(A)$ and $\sum_{i=1}^{n} w_i(OPT)$, we can easily show that for fixed m and d, and an arbitrarily large n, the ratio of the total wait time for the arbitrary algorithm A over the total wait time for the optimal schedule produced by OPT is a bound that approaches $\frac{m}{d}$.

7 Conclusions

In this paper we defined the general problem of on-line data broadcast scheduling. We presented analytical results describing the performance of two well-known scheduling algorithms FCFS and MRF in the context of unit-sized pages and multiple channels. We also established a general lower bound for all on-line algorithms for page selection.

This paper focused on unit-sized pages. An obvious topic for future research is to study the FCFS and MRF algorithms with arbitrary page sizes through competitive analysis. Additionally, other algorithms have been presented [2] that would benefit from similar study. Ultimately, a thorough study of the problem in the context of arbitrary page sizes and multiple channels is desired.

Additional modifications could be made to the model itself. First, assigning a cost for broadcasting each kind of page [6] will result in a more realistic trade-off between quickly satisfying pages and broadcasting a page too frequently (at great expense). Obviously, such an extension to the model will demand more complex scheduling algorithms capable of addressing two minimizing criteria. Also, this paper assumes that the broadcasting of pages cannot be preempted. Relaxing this restriction, particularly in a arbitrary page-size setting, opens several areas of further study. Intuitively, we would expect the use of preemption to lower the total wait time experienced by new on-line algorithms capable of taking advantage of this option in contrast to their non-preemptive counterparts [6]. Finally, each request in the current model arrives with the assumption that the request can be satisfied at any time in the schedule. A more realistic model would include deadlines for each request. If a request is not satisfied before the deadline, it can never be satisfied afterward. Such an extension to the model would also create a new metric of performance as it now allows the possibility that not all requests arriving are satisfied.

References

[1] D. Aksoy, M. Altinel, R. Bose, U. Cetintemel, M. Franklin, J. Wang, and S. Zdonik. Research in data broadcast and dissemination. In *Proceedings of the 1st International Conference on Advanced Multimedia Content Processing*, 1998.

[2] D. Aksoy and M. Franklin. Scheduling for large-scale on-demand data broadcasting. In *Proceedings of the IEEE INFOCOM Conference*, pages 651–659, 1998.

[3] S. Hameed and N. H. Vaidya. Efficient algorithms for scheduling data broadcast. *Wireless Networks*, 5(3):183–93, 1999.

[4] R. M. Karp. On-line algorithms versus off-line algorithms: How much is it worth to know the future? Technical Report TR-92-044, International Computer Science Institution, 1992.

[5] W. Mao. Competitive analysis of on-line algorithms for on-demand data broadcast scheduling. In *Proceedings of the International Symposium on Parallel Architectures, Algorithms, and Networks*, pages 292–296, 2000.

[6] N. Schabanel. The data broadcast problem with preemption. In *Proceedings of the 17th Symposium on Theoretical Aspects of Computer Science*, pages 181–192, 2000.