Competitive Analysis of On-line Algorithms for On-demand Data Broadcast Scheduling

Weizhen Mao
Department of Computer Science
The College of William and Mary
Williamsburg, VA 23187-8795
USA
wm@cs.wm.edu

Abstract

We consider a communication problem, in which requests for data items arrive from time to time via a terrestrial network and at each broadcast tick the single server chooses a data item to broadcast via a satellite downlink to satisfy all the requests for the item. The goal of the server is to minimize the total wait time of the requests. In this paper, we examine two well-known on-line algorithms used by the server to decide which data item to broadcast at each broadcast tick. In particular, we present complete competitive analysis of the algorithms, a research aspect of the problem which had not been studied before. Our results are consistent with observations obtained by simulation experiments reported in past literature. In addition, we prove a general lower bound to the competitive ratios of all on-line algorithms. The competitive ratios obtained for the two on-line algorithms in fact match our general lower bound, indicating the optimality of the algorithms.

1 Introduction

We consider on-demand data broadcast, in which clients send requests for data items (such as a web page or database object) that are not available locally via a terrestrial network, and the single server broadcasts the requested data items in a certain order it chooses via a satellite downlink to satisfy the clients' requests. Such an arrangement is similar to many satellite data services including Hughes Network System's DirecPC architecture [3].

A scheduling problem arises in this situation. Requests for various data items arrive from time to time. Each request is specified by its arrival time and the item requested (e.g., a page number). We say a request is satisfied once the item it has requested is being broadcast. Note that a request can only be satisfied after it arrives. So if a request arrives after the broadcast of the page it happens to request, the same page has to be broadcast again at a later time to satisfy the request. The degree of satisfaction of a request can be measured by its wait time, which is the length of the time period between the arrival time and the time when it is satisfied (or the time broadcast of the requested page starts). At the server's end, a queue is maintained to keep all the requests that have arrived but not yet been satisfied. Assume that each data item is of one unit of length. Therefore, it takes one time unit to broadcast an item. Once the single broadcast channel (the satellite downlink) becomes idle, the server selects a new data item to broadcast. This activity as a result will satisfy all requests for the item waiting in the queue. Deciding which data item to broadcast is in fact a scheduling decision. The server may play fairness by broadcasting the item requested by the earliest arriving request, or it may favor popularity by broadcasting the item requested by the most requests. To reflect the quality of service, the server's goal is to make the total wait time among all requests as small as possible.

Several greedy scheduling algorithms were proposed in [4] and [8] back in the late eighties. More sophisticated algorithms were given in [1], [2], and [3]. All of these algorithms are on-line in the sense that broadcast decisions are made solely based on the requests waiting in the queue without any knowledge of future requests. The analysis of the algorithms, in terms of the quality of broadcast schedules measured by the total (or average) wait time, was done mostly through simulation experiments. The implementation of these algorithms had been considered straightforward until recently new data structures were developed and used in [5] to achieve better time efficiency. One aspect of the algorithms that was not examined is how well they perform as compared to an optimal off-line schedule (computed with knowledge of the future), with respect to a criterion such as minimum total wait time. This is called competitive analysis, commonly used to study on-line algorithms [6] [7]. In fact, competitive analysis was mentioned in [5] as a future research topic for data broadcast scheduling. It motivated our research to be reported in this paper.

We organize the paper as follows. In Section 2, we formally define the problem and describe the environment and assumptions. In Section 3, we describe two previous algorithms, the First Come First Served algorithm (FCFS) and the Most Requests First algorithm (MRF), which make decisions in favor of "cold" and "hot" data items, respectively. In Section 4, we present the competitive analysis of these two algorithms. In Section 5, we give a general lower bound to the competitive ratios for all on-line algorithms for data broadcast scheduling. Finally, in Section 6, we make our conclusions and discuss future research directions.

2 Definitions and Assumptions

Suppose that there are m pages (data items) that will be requested and that each page takes one time unit to broadcast. The unit time period during which broadcast of a page takes place is called a broadcast tick. Suppose that there are n requests all with non-negative integral arrival times and that the pages being requested are all available in the server's database.

The server maintains a queue. Once a request arrives it is placed in the queue immediately. Assume the current time is t, which is the start of the broadcast tick $[t, t+1)$. The server does three things:

1. Request receipt: Add the requests arriving at time t into the queue;
2. Page selection: Choose a page (among those requested by the requests in the queue) to broadcast at the next broadcast tick $[t+1, t+2)$; and
3. Request removal: Delete the requests which will be satisfied by the broadcast of the selected page from the queue.

The time needed to complete the above three steps is usually smaller (in most cases much smaller) than the time needed to broadcast a page. So it is safe for us to assume that when the server finished processing requests and selecting a page, the next broadcast tick $[t+1, t+2)$ has not started yet. So the page selected will be ready for the broadcast tick.

To summarize, at the broadcast tick $[t, t+1)$, there are two kinds of activities taking place simultaneously. The server activity includes adding requests arriving at time t, selecting a page to be broadcast at the next broadcast tick $[t+1, t+2)$, and deleting requests which will be satisfied at time $t+1$. The channel activity includes broadcasting the page selected by the server at the previous broadcast tick $[t-1, t)$. While no algorithm is needed for the channel activity and the two steps of request processing in the server activity, sophisticated algorithms (heuristics) are necessary to help the server choose pages with the goal to minimize the total wait time of the requests. Since a request arriving at time t can only be satisfied no earlier than time $t+1$, its wait time is therefore at least 1, giving a lower bound of n to the total wait time. The scheduling algorithms (which may also be called page selection algorithms) have access to the queue of waiting requests, but have no knowledge about the future requests. Their decisions of which page to broadcast at the next broadcast tick are local and approximate. Which strategies may help produce a solution with a relatively small total wait time close to that of the optimal solution? We will next discuss two page selection algorithms, both of which are based on the greedy method.

3 On-line Algorithms

There are several algorithms for page selection, well-studied in the context of simulation experiments and performance analysis using queueing theory. Two of them are the First Come First Served algorithm (FCFS) and the Most Requested First algorithm (MRF). FCFS always chooses the page of the request with the earliest arrival time. In other words, it focuses on fairness and favors "cold" pages. MRF always chooses the page with the most requests. So it concerns with popularity and favors "hot" pages.

According to [4], when the arrival rate of the requests (which is measured by the number of in-coming requests per broadcast tick) is low, say no larger than 10, FCFS beats MRF by a reasonable margin. Specifically, the average wait time (i.e., total wait time divided by n) of the FCFS solution is about 75% of that of the MRF solution. However, when the arrival rate increases, the difference gets smaller. In fact, when the arrival rate goes beyond 100, the performance of the two algorithms converges to identical. Our analytical results in the following section verify this observation. We will show that in the worst-case both FCFS and MRF produce solutions of total wait time within m times the total wait time of optimal solutions. Recall that m is the number of pages. This type of analysis is called competitive analysis. It is always done to analyze on-line algorithms. In the eyes of theoretical computer scientists, the jigsaw puzzle of an on-line algorithm is never complete without finding the piece representing competitive analysis. This is what we will try to accomplish in the next section.

4 Competitive Analysis

Consider problem instances with m pages and n requests. For request i, assume that the page requested is

p_i and the arrival time is a_i. Let b_t be the page broadcast during $[t, t+1)$ in the schedule obtained by an algorithm. (Whether it is FCFS or MRF will be made clear from the context.) We say that request i is satisfied at time s_i, where s_i is the smallest integer greater than a_i such that $b_{s_i} = p_i$. We then define the wait time of request i to be $w_i = s_i - a_i$. Since $s_i > a_i$, we have $w_i \geq 1$ in any schedule. Therefore, the total wait time of any schedule is at least n. Or equivalently, $\sum_{i=1}^{n} w_i \geq \sum_{i=1}^{n} 1 = n$. So n is a lower bound to the total wait time in any schedule, including the optimal schedule.

In the following subsections, we will present our competitive analysis for the FCFS and MRF algorithms. In particular, we will prove that for any problem instance the total wait time of the solution given by FCFS (or MRF) is never larger than m (also called the competitive ratio) times that of the optimal solution. Clearly, competitive analysis deals with the worst case and may be overly pessimistic. In fact it is possible that for most instances (except for very few), the ratio is much smaller than m. However, the analysis does provide a worst-case guarantee and is an important measure of the overall quality of an on-line algorithm.

4.1 First Come First Served (FCFS)

For any instance, let $\sum_{i=1}^{n} w_i(FCFS)$ be the total wait time of the solution given by FCFS and $\sum_{i=1}^{n} w_i(OPT)$ be the total wait time of the optimal solution. Let m be the number of pages.

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

Proof We already know that $\sum_{i=1}^{n} w_i(OPT) \geq n$. If we can show that $w_i \leq m$ for $i = 1, \ldots, n$ in the FCFS solution, then we have

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

By contradiction, assume that there is a request, say i, with $w_i > m$. In other words, the request arrives at time a_i and is satisfied at time $a_i + w_i$. Since $w_i > m$, between $a_i + 1$ and $a_i + m$ page p_i is never broadcast. During this period, there are m broadcast ticks, $[a_i+1, a_i+2), \ldots, [a_i+m, a_i+m+1)$, to broadcast pages chosen from $m-1$ pages (excluding page p_i). Note that the broadcast channel stays busy during this period since otherwise page p_i would have been chosen to broadcast to fill the idle tick, which would result in $w_i \leq m$. By the pigeonhole principle, at least one page, say p, among the $m-1$ pages must have been broadcast more than once. The second broadcast of the page satisfies requests all of which have arrival times no earlier than $a_i + 1$ (while the first broadcast satisfies those requests with arrival times earlier than $a_i + 1$). This is in contradiction to the FCFS algorithm – request i arrived at a_i so page p_i should have been chosen to broadcast instead of page p being broadcast the second time. To conclude, all requests have wait time no larger than m in any FCFS solution.

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

Proof Consider the following instance with $n = lm$ for some l. Assume that for request $i = km + j$, where $j = 1, \ldots, m$ and $k = 0, \ldots, l-1$, the requested page is $p_i = j$ and the arrival time is $a_i = 0$ if $k = 0$ and $a_i = (k-1)m + j$ if $k \geq 1$.

The FCFS algorithm gives a schedule which broadcast pages in a circular order of $1, 2, \ldots, m$ until all n requests are satisfied, i.e., the page broadcast at time t, $b_t = j$, where j is the integer between 1 and m such that $t = km + j$ for $k \geq 0$. Now consider the wait times of the requests. We have $w_1 = 1, w_2 = 2, \ldots, w_m = m$ for the first m requests. For the remaining $n - m$ requests, $w_i = m$. Therefore,

$$\begin{aligned}\sum_{i=1}^{n} w_i(FCFS) &= (1 + 2 + \cdots + m) + (n-m)m \\ &= nm - \frac{1}{2}m(m-1).\end{aligned}$$

The optimal schedule will have unit wait time for most of the requests. Specifically, $w_1 = 1, w_2 = 3, w_3 = 4, \ldots, w_m = m+1$ for the first m requests and $w_i = 1$ for the remaining $n - m$ requests. Therefore,

$$\begin{aligned}\sum_{i=1}^{n} w_i(OPT) &= (1 + 3 + 4 + \cdots + (m+1)) \\ &\quad +(n-m)1 \\ &= n + \frac{1}{2}(m+2)(m-1).\end{aligned}$$

Therefore, for fixed m and arbitrarily large n, the ratio of $\sum_{i=1}^{n} w_i(FCFS)$ over $\sum_{i=1}^{n} w_i(OPT)$ approaches to m.

4.2 Most Requested First (MRF)

For any instance, let $\sum_{i=1}^{n} w_i(MRF)$ be the total wait time of the solution given by MRF and $\sum_{i=1}^{n} w_i(OPT)$ be the total wait time of the optimal solution. Let m be the number of pages.

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

Proof We define the following notation. Let A_t be the number of requests arriving at t. Let S_t be the number of requests satisfied by the page broadcast at t, which is b_t. Let W_t be the number of requests still waiting at t. Let k be the last broadcast tick in the MRF schedule. Observe that $W_k = 0$, $A_k = 0$, $S_k > 0$, and $\sum_{t=1}^{k} S_t = \sum_{t=0}^{k-1} A_t = n$. Because MRF always chooses the page requested most, S_t is at least as large as the number of requests for any page other than page b_t. So $(m-1)S_t \geq W_t$. We have

$$\sum_{t=1}^{k-1} W_t \leq (m-1)\sum_{t=1}^{k-1} S_t < (m-1)\sum_{t=1}^{k} S_t = (m-1)n.$$

Consider the total wait time $\sum_{i=1}^{n} w_i(MRF)$. A_0 is the amount of wait time accumulated during $[0,1)$. $W_1 + A_1$ is the amount of wait time accumulated during $[1,2)$. $W_2 + A_2$ is the amount of wait time accumulated during $[2,3)$. And finally, $W_{k-1} + A_{k-1}$ is the amount of wait time accumulated during $[k-1,k)$. So we have

$$\begin{aligned}\sum_{i=1}^{n} w_i(MRF) &= A_0 + (W_1 + A_1) + (W_2 + A_2) \\ &\quad + \cdots + (W_{k-1} + A_{k-1}) \\ &= \sum_{t=0}^{k-1} A_t + \sum_{t=1}^{k-1} W_t \\ &< n + (m-1)n \\ &= mn\end{aligned}$$

Together with $\sum_{i=1}^{n} w_i(OPT) \geq n$ for the optimal algorithm, we get $\sum_{i=1}^{n} w_i(MRF) \leq m \sum_{i=1}^{n} w_i(OPT)$.

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

Proof Interestingly, the instance used for the FCFS proof (Theorem 2) can be used to yield the same result for the MRF algorithm. Note that when MRF is applied to the instance, every page selection decision is made breaking an m-way tie in favor of the page requested by the earliest arriving request.

5 A General Lower Bound

In this section, we will prove that no on-line algorithm for page selection has a competitive ratio better than m, the number of pages. This indicates that the on-line algorithms we have studied in this paper are optimal, with competitive ratios matching the general lower bound.

THEOREM 5 *For any on-line algorithm for page selection, its competitive ratio is at least m, where m is the number of pages.*

Proof Using an adversary argument, we assume that the input instance is provided by an adversary. Since the arbitrary scheduling algorithm we consider is on-line, it does not have access to the entire request sequence at once, but instead it only sees the requests that have arrived and has to make page selection decisions without any knowledge of future requests. Because of this limitation of the on-line algorithm, the adversary is able to give out the request sequence little by little, making adjustments according to the decisions the algorithm has made, as long as it seems consistent in the eyes of the on-line algorithm.

To begin, the adversary first give out m requests with $a_i = 0$ and $p_i = i$ for $i = 1, 2, \ldots, m$. The on-line algorithm then selects one of the m pages to broadcast at time 1, say, it is page b_1. From then on, the adversary will generate one request at each broadcast tick, asking for the page that is just being chosen. To be more specific, request $m+i$ arrives at time i and asks for page b_i, for $i = 1, 2, \ldots, n-m$. It results in a schedule in which for each page there is exactly one request waiting at any time between 1 and $n-m$. After time $n-m$ no more requests will come and each of the m pages have to be broadcast one more time to satisfy all the waiting requests. Let A denote the on-line algorithm. Considering the total wait time, we notice that for page b_{n-m+i}, with $i = 1, 2, \ldots, m$, the total wait time of all the requests for the page is $n - m + i$. Summing up, we have

$$\begin{aligned}\sum_{i=1}^{n} w_i(A) &= (n-m+1) + (n-m+2) + \cdots \\ &\quad +(n-m+m) \\ &= mn - \frac{1}{2}m(m-1).\end{aligned}$$

Next we will argue that for the same input instance there is a schedule in which $w_i = 1$ for most requests. Studying the input provided by the adversary, we notice that there are $2m-1$ requests arriving before time m, among which m requests arrive at time 0 and $m-1$ requests arrive at time instants $1, 2, \ldots, m-1$. We will construct a schedule that broadcasts pages to satisfy these $2m-1$ requests at or before time m. From time m and beyond, since there will be just one request arriving at each time instant, it can be easily satisfied after just one time unit of minimum waiting. Thus, w_i would be 1 for $i = 2m, 2m+1, \ldots, n$.

We now focus on how to satisfy the first $2m-1$ requests by time m. We define $R_p = \{i | a_i < m \text{ and } p_i = p\}$ for $p = 1, 2, \ldots, m$. So R_p is in fact the set of requests which arrive before time m and request for page p. Obviously, $|R_p| \geq 1$ for $p = 1, 2, \ldots, m$. For R_p with more than one request, let $l(p)$ be the request with the latest arriving time among all in R_p. We define a schedule that broadcasts page p at time $a_{l(p)} + 1$ for p with $|R_p| \geq 1$ and broadcasts page p at any of the remaining idle ticks (at or before m) for p with $|R_p| = 1$. Note that since there is only one request arriving at each tick after 0 and before m the schedule just described will never try to broadcast two different pages at the same broadcast tick. It is not very hard to see that for those m requests arriving at time 0, their total wait time is $1+2+\cdots+m = \frac{1}{2}m(m+1)$, and for each of the remaining $m-1$ requests arriving between time 1 and time $m-1$, its wait time is at most $m-1$, with the total wait time to be at most $(m-1)^2$. Considering all n requests, we have

$$\begin{aligned}\sum_{i=1}^{n} w_i(OPT) &\leq \frac{1}{2}m(m+1) + (m-1)^2 \\ &\quad +(n-2m+1)1 \\ &= n + \frac{1}{2}(m-1)(3m-4).\end{aligned}$$

Therefore, for fixed m and arbitrarily large n, the ratio of $\sum_{i=1}^{n} w_i(A)$ over $\sum_{i=1}^{n} w_i(OPT)$ is at least a bound that approaches to m. Since algorithm A is arbitrary throughout the proof, such a lower bound holds for all on-line algorithms for page selection.

6 Conclusions

In this paper, we gave competitive analysis of two simple on-line algorithms for page selection in on-demand data broadcast. We proved that for both the FCFS and MRF algorithms the competitive ratio is m and that this bound is tight. Our competitive analysis filled a gap that had been ignored in past research. Furthermore, it verified previous simulation results in a rigorous mathematical manner. We also proved that no on-line algorithm for page selection has a competitive ratio better than m. This suggests that both the FCFS and MRF algorithms are optimal in the worst case.

There are two other algorithms which showed much better performance in simulation. They are the Longest Wait First algorithm (LWF) and the Requests Times Wait algorithm (RxW). The algorithms are more sophisticated – even their implementations require complex data structures [5]. An obvious topic for future research is to study the LWF and RxW algorithms through competitive analysis. Using the general lower bound result in this paper, we know that the competitive ratio is at least m for both algorithms. So the next immediate step is to determine upper bounds of the competitive ratios.

Acknowledgment

This research was supported by a Faculty Research Assignment grant awarded by the College of William and Mary and was partially done when the author was on leave at Princeton University.

The author also wishes to thank Dr. Robert E. Tarjan for bringing the problem of data broadcast scheduling to her attention.

References

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

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

[3] D. Aksoy and M. Franklin, On-demand broadcast scheduling, manuscript, 1999.

[4] H. D. Dykeman, M. Ammar, and J. W. Wong, Scheduling algorithms for videotex systems under broadcast delivery, in the *Proceedings of the IEEE International Conference on Communications*, 1847–1851, 1986.

[5] A. V. Goldberg, H. Kaplan, and R. E. Tarjan, Heaps with times, applied to broadcast scheduling, manuscript, 1999.

[6] S. Irani and A. R. Karlin, Online computation, in *Approximation algorithms for NP-hard problems*, Chapter 13, edited by D. S. Hochbaum, PWS Publishing Company, Boston, 1997.

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

[8] J. W. Wong, Broadcast delivery, *Proceedings of IEEE*, 76 (12), 1566-1577, 1988.