A Heuristic Approach for Constructing Improved Broadcast Schedules

Stathis B.Mavridopoulos, Petros Nicopolitidis, *Member, IEEE*,
Georgios I.Papadimitriou, *Senior Member, IEEE*

***Abstract*—This paper proposes a heuristic algorithm for constructing improved broadcast schedules. With simple modifications to the well-known Broadcast Disks approach, the proposed technique provides increased performance. This is confirmed by simulation results.**

I. Introduction

THERE are two technologies for wireless data broadcasting systems: pull-based (e.g. [1-3]) and push-based (e.g. [4-8]). In the former clients "pull" data from the server in order to provide data to locally running applications. Pull-based systems are poor match for asymmetric communication environment though, because they require substantial upstream communications capabilities. In those environments asymmetry arises from the fact that the downstream capacity to clients from servers is much greater than the upstream communication capacity. A push-based architecture exploits the relative abundance of downstream communications capacity in asymmetric environments. In push based architecture, data is pushed continuously and repeatedly from the server out to the clients.

Broadcast Disks [5] is a push-based architecture for data broadcasting. However, it has been reported that in certain cases a flaw in its logic significantly degrades its performance. Our proposal is a simple heuristic that improves the performance of Broadcast Disks in such cases.

Throughput this paper, we focus on the construction of the broadcast program at the server side. So, we consider clients that do not have any cache or memory. The data access pattern for the clients and their population is static. The data access probabilities are known in the environment and they do not change. Data items are of uniform, fixed length, which we call pages for simplicity, and they do not change during the broadcast.

The simplest way to transmit data is the "flat broadcast", which is schematically shown in Figure 1a. When a client needs a data item, it monitors the broadcast and waits for the item to arrive. In flat broadcast, all data items or "pages" are been transmitted evenly and cyclicly. The expected delay for a page on the broadcast is the same for all pages.

If we transmit a certain page from a flat broadcast with different frequency, then that page will be received faster from

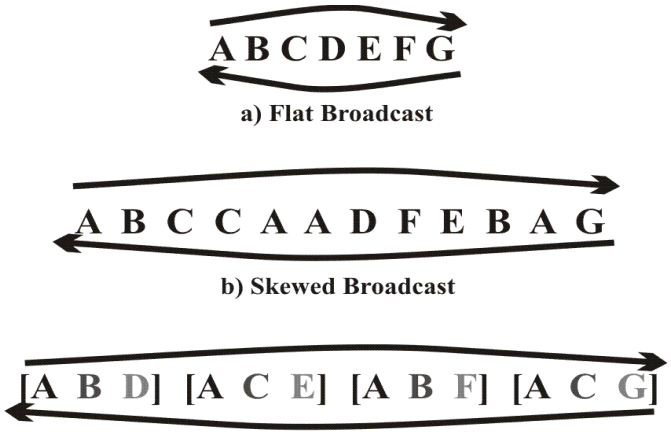

Fig. 1. Flat, Skewed and Broadcast-Disks compatible schedules

the clients that need that certain page, however we increase the cycle of the broadcast, delaying all other pages. So, a flat broadcast is only ideal for uniformly accessed data. In most systems though, data is been requested in a non-uniformly, random and skewed way. The flat broadcast becomes less effective as the skewness of the access probabilities increase. For simplicity, we already assumed that we know the data access probabilities for the client population and those access probabilities do not change. Then, one would suggest to broadcast the pages in frequencies accordingly to their relative access probabilities. So, if page A is three times more likely to be requested than page B, then we transmit page A three times while we transmit page B one time in every cycle. The broadcast cycle can be constructed by randomly placing the pages. We will refer to that logic, which is schematically shown in Figure 1b, as skewed broadcast. It can be easily proved [4] that the skewed broadcast is more effective than flat broadcast, but only when the access probabilities are greatly skewed.

Another advancement in terms of performance can be given by the Broadcast Disks approach. The reason Broadcast Disks performs better than skewed broadcast can be understood through the so-called Bus Stop Paradox. If the inter-arrival rate of Buses is fixed, e.g. buses arrive every 30 minutes, then the expected delay for a passenger arriving at a bus stop at a random time will be 15 minutes, which is the average of the time between buses' arrivals. If however buses arrive according to a Poisson process (completely randomly) at the bus-stop on average every 30 minutes, then the passenger will have to wait for 30 minutes, because Poisson arrivals are memoryless and thus the time until the next bus is independent of how long it has been since the last bus. So, the skewed broadcast, where pages arrive completely randomly,

S.B.Mavridopoulos, P. Nicopolitidis, G. I.Papadimitriou are with the Department of Informatics, Aristotle University of Thesssaloniki, Box 888, 54124, Thessaloniki, Greece (e-mails: stathism@csd.auth.gr, petros@csd.auth.gr, gp@csd.auth.gr)

978-1-4244-8489-8/10/$26.00 © 2010 IEEE

could be improved by uploading data in a certain manner; that is the logic behind Broadcast Disks.

In Broadcast Disks program construction, which is schematically shown in Figure 1c, the broadcast is created by assigning data items to disks of varying sizes and speeds, and then multiplexing the disks on the broadcast channel. Items stored on faster disks are broadcast more often than items on slower disks. In this approach the inter-arrival rate of disks/pages is fixed, in contrast to the skewed broadcast, thus decreasing the expected delay as the variance in inter-arrival rate decreases.

The authors in [4] describe the broadcast program generation in the following steps:

1. Order the pages from hottest (most popular) to coldest.
2. Partition the list of pages into multiple ranges, where each range contains pages with similar access probabilities. These ranges are referred to as disks.
3. Choose the relative frequency of broadcast for each of the disks. The only restriction on the relative frequencies is that they must be integers.
4. Split each disk into a number of smaller units. These units are called chunks (*Cij* refers to the jth in disk *i*). First, calculate max_chunks as the Least Common Multiple (*LCM*) of the relative frequencies. Then, split each disk *i* into *num_chunks (i) = max_chunks / rel_freq (i)* chunks.
5. Create the broadcast program by interleaving the chunks of each disk in the following manner:

```
01 for i := 0 to max_chunks -1
02      for j:= 1 to num_disks
03              Broadcast chunk Cj, (i mod num_chunks (j))
04      endfor
05 endfor
```

II. Motivation for the Proposed Approach

A. When Broadcast Disks underperform

Authors in [10, 11] describe and prove the conditions that should be avoided in order to keep the expected delay for all data items small:

- The number of disks and *LCM* of all relative frequencies must be kept small, otherwise the program period will increase significantly making bigger average delays on all data-items.
- The number of chunks and the number of pages assigned to each disk should be divisible, otherwise null pages are produced.

It is very difficult to choose relative frequencies because they must satisfy several divisibility conditions for keeping the *LCM* as low as possible and at the same time they must satisfy other divisibility conditions for keeping the number of items divisible with the number of chunks.

Theoretically a system admin could carefully select all the parameters and create an optimal broadcast program. In many applications, however, developers or system managers cannot always choose the best conditions. Or maybe the broadcast is created automatically, which is a more realistic scenario in real, ever-changing world. So, we need an algorithm that would generate a good broadcast no matter what the parameters are.

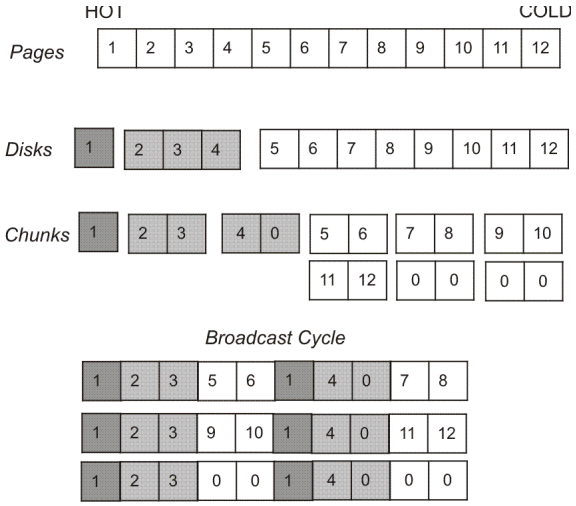

Fig. 2. Broadcast program construction via Broadcast Disks. Different colors mean different disks.

B. Broadcast Disks Program Construction Example

In order to demonstrate the broadcast program generation algorithm and point its flaw, we will use an example, which is schematically shown in Figure 2.

Consider a 12 pages example, where we can recognize 3 ranges/disks. Items in disk 1 are been requested twice times as items in disk 2 and six times as frequently as in disk 3. Thus, relative frequency of disk 3 is 1 (or *rel_freq*(3) = 1), *rel_freq*(2) = 3 and *rel_freq*(1) = 6.

The *LCM* is 6 and the number of chunks for each disk is (Figure 2):

Num_chunks(1)=6/*rel_freq*(1)=1
Num_chunks(2)=6/*rel_freq*(2)=2
Num_chunks(3)=6/*rel_freq*(3)=6

Unfortunately, disk 2 and disk 3 cannot be divided in the required number of chunks without the addition of null pages. However in a broadcast of 30 pages the 7 null pages consist the 23% of the broadcast. The more we increase the length of the major-cycle, the more we increase the expected delay, making broadcast disks even slower than flat broadcast.

As can be seen in the example previously, the broadcast program generation produces *null* pages (marked with zero) in Step 4 of the Broadcast Disks program generation algorithm, when the disk cannot be evenly divided. While authors in [4] believe that it is most important to keep the periodicity of the system by keeping those null pages or transmit essential data or even broadcast extremely important pages in place of the null pages, our tests indicate that those extra pages burden the broadcast considerably.

III. The Proposed Scheme

A. Shift Null Pages

What we suggest is to "shift" the broadcast program over the null pages, basically to remove those pages from the broadcast. This greatly improves performance, with one drawback: we lose the periodicity of the broadcast. This procedure is schematically shown in Figure 3b. Even when we insert skewness in the broadcast, the removal of the null pages greatly increases performance.

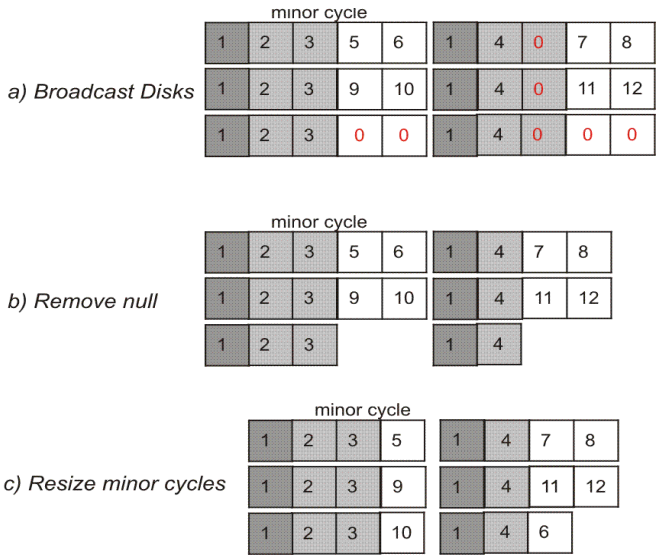

Fig. 3. Proposed Modifications.

B. *Resize Minor Cycles*

However, we can further improve the broadcast program by restoring the order, decreasing the randomness we created in the length of the minor-cycles. By minor-cycle, we mean the set of *Nth* chunks that are transmitted from every disk (the inner-loop, Step 5 of broadcast generation program).

More null pages will appear in some minor-cycles than others and removing those null pages will affect the length of those cycles. Consequently we insert skewness into the broadcast; skewness is what Broadcast Disks is trying to avoid, though.

A simple way to solve this is to transfer pages from large minor-cycles to smaller ones, thus creating a broadcast with minor-cycles with equal lengths. Tests indicate that performance will further increase by applying this method, which is schematically shown in Figure 3c.

To sum up, what we propose is to keep the broadcast generation algorithm from Broadcast Disks as it is and then apply the methods above to improve its performance:

- Generate Broadcast Program
- Remove null pages
- Resize minor-cycles

IV. Simulation Results

A. *Environment*

In order to test our theory and export results, we used simulation programs to test the broadcast for all different broadcast systems. The program demonstrates a client and a server, the server broadcasts pages based on the broadcast algorithm and the client requests pages based on the access patterns we set each time. Then, we can calculate the mean delay between successful arrivals of pages so we can directly compare each method.

The access patterns were chosen from Zipf distribution, one of a family of related discrete power law probability distributions. This distribution has a parameter a>1 and is defined by the probabilities [6]:

$$p_i = \frac{1}{\varsigma(\alpha) i^{\alpha}}, (i \geq 1),$$

where $\varsigma(\alpha) = \sum_{i=1}^{\infty} \frac{1}{i^{\alpha}}$ is the Riemann Zeta function. For $a \rightarrow 1$, the produced distribution is increasingly skewed. While a increases, the distribution becomes more uniform like.

The disks relative frequencies are calculated using the Δ formula, as been described in [4]:

$$\frac{rel\_freq(Di)}{rel\_freq(Dn)} = (N - i)\Delta + 1$$

Where $rel\_freq(Di)$ {i=1,…,N}, is the relative frequency of Disks I, and the frequency of the slowest disk is selected $rel\_freq(Dn) = 1$.

Another aspect of the broadcast is the way of separating the data items into disks. In our experiments we used the rule that each disk should have twice the amount of items of the previous one. So if D1 had 50 pages, D2 would have 100, D3 200 pages etc. For a different setup the results might slightly differ.

For simplicity we will use symbols every time we refer to parameters in the results section:

M the number of pages that will be broadcasted
N the number of different broadcast disks
a the zipf's parameter
Δ the broadcast parameter Delta

Our results were taken by varying a parameter each time. Average Delay is measured in broadcast units and is characteristic of the performance of the broadcast.

B. *Results*

Varying a Parameter

In this experiment we chose: M = 5000, N = 5, a = 1.6..3.4, Δ = 3

While varying the parameter a, we tested the broadcast for different data access patterns. a = 1.6 means a greatly skewed distribution set, while a = 3.4 means almost uniformly accessed data. The graph in Figure 5, demonstrates the benefits of our proposed approach over Broadcast Disks in terms of average delay for an arbitrary item. It can be seen that our approach significantly outperforms that of Broadcast Disks by providing a lower delay over the values of parameter a.

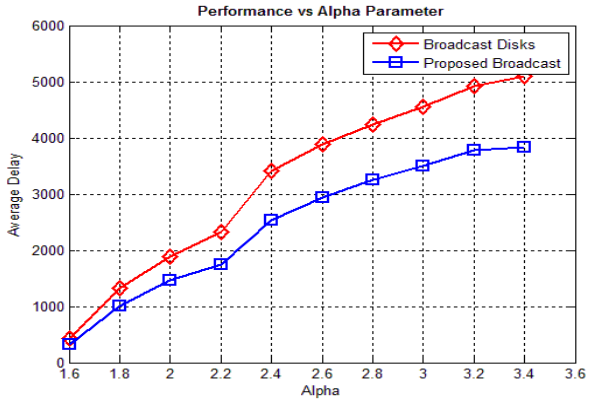

Fig. 4. Average delay for the Broadcast disks and the proposed approach versus skew parameter *a*.

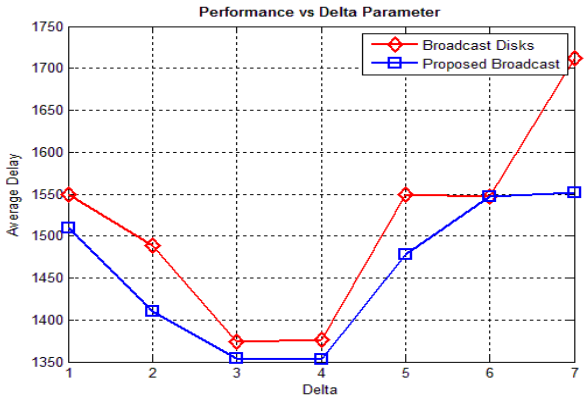

Fig. 5. Average delay for the Broadcast disks and the proposed approach versus parameter Δ.

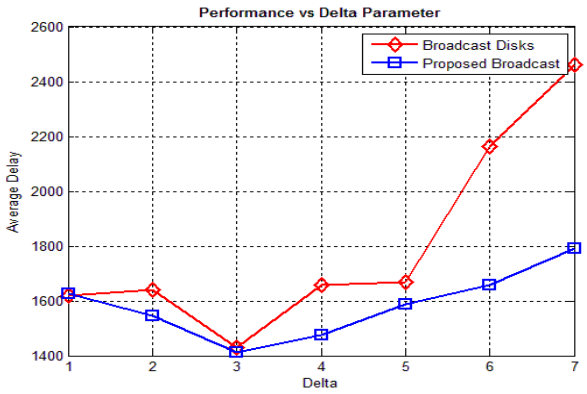

Figure 6. Average delay for the Broadcast disks and the proposed approach versus parameter Δ using another sample.

Varying Delta(Δ) Parameter

In this experiment, whose results are shown in Figure 6, we chose: $M = 5000$, $N = 3$, $a = 2$, $\Delta = 1..7$. We observe that the broadcast performs best at $\Delta = 4$. For larger or smaller Δ, both Broadcast Disks and the proposed algorithm perform worse. The optimal Δ is highly depended on the selection of a parameter and the method of creating the disks, as already confirmed in related research [4]. It can be seen that our approach significantly outperforms that of Broadcast Disks by providing a lower delay over the values of parameter Δ.

Similar conclusions can be drawn for other environments. For example, in Figure 7 we present results for $M = 5000$, $N = 4$, $a = 2.25$, $\Delta = 1..7$. Similar results to Figures 6 and 7 will be reported by varying any other parameter. The proposed algorithm will always produce shorter average response times than Broadcast Disks.

Varying the number of Disks(N) parameter

In this experiment group we chose: $M = 5000$, $N = 1..9$, $a = 2.5$, $\Delta = 1$.

As stated in [10, 11] the number of disks should be kept small, otherwise the broadcast underperforms. However, the proposed algorithm broadcast is not affected negatively, in fact it benefits when more disks are used. The results presented in Figure 8 confirm this statement

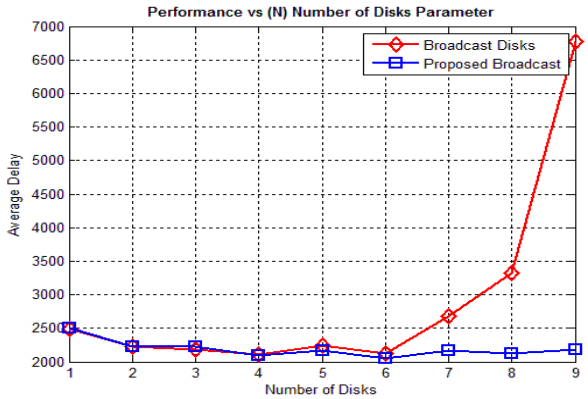

Fig. 7. Average delay for the Broadcast disks and the proposed approach versus the number of disks N.

V. Conclusion

This paper focused on the well known Broadcast Disks technique. Broadcast Disks could perform optimal if all the parameters are chosen carefully; otherwise the generated broadcast could suffer. This paper proposes a heuristic algorithm for constructing improved broadcast schedules. With simple modifications to the well-known Broadcast disks approach, the proposed technique provides increased performance. This is confirmed by simulation results. Authors in [10, 11] proposed solutions to the same problem; basically their generated broadcast and ours are largely similar, and similar are our results. However, the logic we followed and the method we proposed are simpler and easier to implement.

References

[1] D. Aksou and M. Franlkin, "Scheduling for large-scale on-demand data broadcasting," in Proceedings of IEEE INFOCOM, 1998, pp. 651–659.

[2] K. Stathatos, N. Roussopulos, and J. S. Baras, "Adaptive data broadcast in hybrid networks," in Proceedings of VLDB, 1997, pp. 326–335.

[3] S.Acharya, M.Franklin, S.Zdonik, Balancing Push and Pull for Data Broadcast, in Proceedings of ACM SIGMOD, May, 1997, pp. 183-194.

[4] S.Acharya, M.Franklin, S.Zdonik, Dissemination-based Data Delivery Using Broadcast Disks, IEEE Personal Communications, vol. 2, no. 6, pp. 50-60, December 1995.

[5] N. H. Vaidya and S. Hameed, "Scheduling data broadcast in asymmetric communication environments," in ACM/Baltzer Wireless Networks, 1999, vol. 5, pp. 171–182.

[6] P.Nicopolitidis, G.I.Papadimitriou and A.S.Pomportsis, "Using Learning Automata for Adaptive Push-Based Data Broadcasting in Asymmetric Wireless Environments", IEEE Transactions on Vehicular Technology, vol.51, no.6, November 2002, pp.1652-1660.

[7] V.Kakali, G.I.Papadimitriou, P.Nicopolitidis, and A.S.Pomportsis, "A New Class of Wireless Push Systems", IEEE Transactions on Vehicular Technology, vol.58, no.8, October 2009, pp.2529-4539.

[8] P.Nicopolitidis, G.I.Papadimitiou and A.S.Pomportsis, "Continuous Flow Wireless Data Broadcasting for High-Speed Environments", IEEE Transactions on Broadcasting, vol.55, no.2, June 2009, pp.260-269.

[9] Luc Devroye, "Non-Uniform Random Variate Generation", Springer, 1986, pages 550-551.

[10] E.Tiakas, S.Ougiaroglou, P.Nicopolitidis, "Efficient Broadcast Disks Program Construction in Asymmetric Communication Environments", in Proceedings of the 10th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (IEEE/ACM MSWIM07), October 22-26, 2007, Chania, Greece, pp.279-283.

[11] E.Tiakas, S.Ougiaroglou and P.Nicopolitidis, "Efficient Algorithms for Constructing Broadcast Programs in Asymmetric Communication Environments", Telecommunication Systems, Springer, vol.21, no.3, 2009.