A New Approach to the Design of Wireless Data Broadcasting Systems: An Analysis-Based Cost-Effective Scheme

Christos K. Liaskos, *Member, IEEE,* Sophia G. Petridou, *Member, IEEE,*
Georgios I. Papadimitriou, *Senior Member, IEEE,*
Department of Informatics
Aristotle University of Thessaloniki
Thessaloniki, Greece 54124
Emails: {cliaskos, spetrido, gp}@csd.auth.gr

***Abstract*—A new, cost aware approach to the design of wireless data broadcasting systems is introduced. The proposed approach is based on the mathematical analysis of the systems' performance optimization, which form their broadcast schedule in accordance with the well-known Broadcast Disks method. The new scheme is characterized by low memory requirements and computational overhead. The computational complexity of the proposed scheme is evaluated and it is found to be lower than the well-known Thomas method-based scheme. Furthermore, the relation between the system's optimal parameters and the number of total broadcasted data objects is being studied through a specially adapted multivariate illustration technique and it is observed to be non-linear.**

***Keywords*-Analytical approach; broadcast disks; low cost; wireless push system.**

I. Introduction

Recent years have witnessed the widespreading use of wireless push-based systems. Simple in architecture and implementation, lightweight and energy efficient - especially from the client's point of view and both hardware and software wise - the push-based approach has been adopted for use in a variety of information dissemination applications and has been incorporated in almost every single mobile telecommunications device. Popular uses include airport and hospital informative systems, instant messaging services, and multimedia on-demand over the internet or cellular networks. Consequently, the ongrowing interest of the telecommunications industry has spurred the research on the resource efficiency and performance optimization of these systems.

The objective of the related research is to define an algorithm that receives as an input the popularity measure of each to-be-broadcasted piece of information, and produces a broadcast schedule that minimizes:

1) The mean waiting time of the wireless clients.
2) The overall implementation cost of the system, which is mainly a function of the computational and memory requirements of the central broadcast server.

It must be stated that when discussing an application of a push-based system in the real world, the latter criteria-i.e the overall cost-is of utmost importance. It essentially defines wether the push-based approach will be adopted instead of a classic pull based scheme. A push-based architecture is preferred as an inexpensive solution and is employed when implementation cost is more important than the overall performance. Pull systems on the other hand perform undoubtedly better but tend to require more powerful-and therefore more expensive-hardware. It is therefore commendable to assume that hardware limitations do exist in push-based systems and every broadcast-related algorithm should pursue minimal computational and memory requirements.

A. Related Work

To the best of our knowledge, the cost aspect of push-based systems has been so far overlooked in the related research, including the authors' own work. On the other hand, much work has been done regarding the minimization of the mean waiting time of the wireless clients:

Ideally, a straightforward mathematical analysis of the system would evince this procedure, yet it has been proven that this approach falls into the NP-hard category of problems [1–3], rendering it impossible to solve with the current mathematical tools and methods.

Less constricting approaches were developed to facilitate research, one of the most popular being the Broadcast Disks method [4]. This method provides a lax broadcast scheduling framework that ensures the periodicity and proportionality attributes of the broadcast program. In other words, it provides a way to keep the time between consecutive broadcasts of the same data item constant, and its total appearances proportional to its popularity. Other than that, it leaves ample room for performance optimization. In its vast majority, research in the field has inclined towards heuristic methods [5–7]. The work of [8] introduced a mathematical analysis of the broadcast disks method and, based on it, provided an optimal broadcast scheduling procedure, which outperformed the previous schemes in the majority of cases (87% out of a total of 54 test cases). In more detail, it resulted in a mathematical equation representing the estimated mean client waiting time as a function of the model-related parameters. Finding the

9781-4244-3941-6/09/$25.00 © 2009 IEEE

minimum value of this function was then a trivial task.

The next step was to use the analysis' results in order to check a plethora of client cases, using the popular parametric client model firstly introduced in [4], thusly overcoming a major omission of related research: newly introduced algorithms were checked against only few or even just one single client case. Checking a wide range of client cases involved the definition of the optimal broadcast parameters through the preceding mathematical analysis, calculating the estimated mean client waiting time and comparing its accordance with corresponding simulation results. This brought to light a major problem: the server's and clients' computational power and memory needs were not taken into consideration.

B. Contribution

Purpose of the present work is primarily to reestablish the cost criteria as a determinative factor of the design of push-based systems. To this end, an improved version of the procedure of [8] will be introduced, in terms of computational and memory needs. As described in Section II, the core of the optimization procedure is the solution of a system of linear equations. By exploiting its unique characteristics a fast optimization algorithm was developed, minimizing the computational requirements of this task. Furthermore, the exploitation of the periodicity and the round robbin manner in which the broadcast schedule is constructed (in accordance with the broadcast disks method) resulted into the development of an extremely memory conserving broadcast schedule constructor algorithm. Comparison with other classical approaches demonstrated the superiority of both of the new algorithms.

This cost reduction had a direct impact on the required simulation time, enabling the testing thousands of client cases in relatively small amount of time. The optimization-based procedure for broadcast scheduling was found to be dominant over other heuristic-based algorithms in all cases. The validity of the analysis was ensured through the accordance of the estimated and simulated mean waiting times. Finally, through a specially adapted multivariate illustration technique, important conclusions were extracted concerning the linearity of the system's behavior regarding the optimal broadcast parameters.

C. Roadmap

The remainder of this paper is organized as follows: in Section II the network architecture and operation as well as the Broadcast Disks Method and its mathematical analysis are described in more detail. In Section III the fast optimization algorithm and memory efficient broadcast schedule constructor algorithm are presented. Section IV refers to the comparison of the algorithms with other related approaches. Section V provides a brief overview of the simulated client cases and presents the specially adapted multivariate illustration technique used to depict the corresponding optimal scheduling parameters. Finally, conclusions are given in Section VI.

Table I
TERMINOLOGY OF PUSH-BASED SYSTEMS.

Term	Description
$page$	Alias for data item. The elementary data unit.
$dbsize$	The number of available pages contained in the broadcast server's database.
$range$	The clients access this many pages from the total, *dbsize* available ones.
$region$	A group of pages that share the same probability of being requested by a client.
$regionsize$	The number of pages in one region.

II. SYSTEM BASICS

A. Network Architecture and Probabilistic Modeling

The network under discussion is considered to be a typical push-based system consisting of a server, a database, one or more clients and an asymmetric communication channel. In general, data obtained from the database is pushed from the server to the clients through the asymmetric channel, which implies that the downstream communication bandwidth from server to clients is much greater than the upstream communication bandwidth from clients back to server [9].

According to the most commonly used probabilistic model [4] of the system, the database contains $dbsize$ equally sized items, also known as pages. The server acts as a cental node of a cellular network [10, 11], while any number of portable devices can act as clients. As in [4] though, the approach of a single, probabilistically equivalent client is adopted. This client accesses only a random portion of the total $dbsize$ available pages, which is denoted as $range$. Thus it holds that $range \leq dbsize$. These pages are organized in k equally sized groups called $Regions$, where $k = \lceil \frac{range}{regionsize} \rceil$, while $regionsize$ denotes the number of pages contained in a single region r_j, $j = 1, \ldots, k$. All $regionsize$ pages in r_j share the same probability of being requested by a client (Table I). The $Regions$ themselves are considered to follow the zipf probability distribution. Thus if all $Regions$ are sorted in a descending fashion according to their request probability, any page in region r_j will have:

$$P(r_j) = \frac{c}{j^\theta}, j = 1, \ldots, k,$$

request probability, where θ is the zipf p.d.f. parameter (and as such $\theta \neq 1$) and c is a constant value conforming to the condition:

$$\sum P(r_j) = 1$$

Thus the client's p.d.f. P_p has the form of Fig. 1. The server gradually obtains an approximate form of P_p by means of an appropriate feedback mechanism [12]. In the

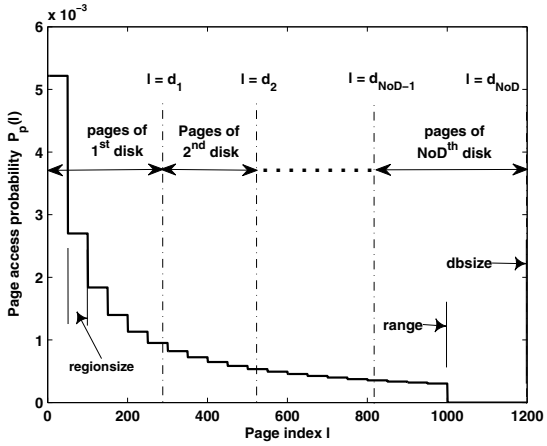


Figure 1. The client's probability distribution function

context of this paper, every simulation that took place was executed under the assumption that enough time has elapsed for the server to have an accurate approximation of P_p as depicted in Fig. 1. The mathematical analysis though is totally p.d.f. agnostic.

B. The Broadcast Disks Method

Once the server has acquired sufficient knowledge of client's P_p the Broadcast Disks method intervenes and acts in three phases:

1) The pages are grouped according to their access probability in a fixed number of groups called disks. These scheme can be abstractedly represented as an array of physical disks spinning around a common axis.
2) Each of these disks is then set to spin with a angular velocity proportional to the aggregate demand of its contained pages.
3) An imaginary set of stationary heads then retrieves pages from the disks and forwards them to the broadcasting system in the same order that they have been read.

Thus the periodicity and proportionality criteria of the broadcast schedule mentioned in Section I are met.

It becomes evident that the broadcast disks method requires a way of defining the number of disks of step 1 and the disk angular velocities of step 2. To this end, in [8] an optimal such procedure was presented, the performance of which was superior to the compared alternatives. Since in the context of this paper this procedure will be altered to require less computational power and memory, it is imperative that a brief overview be given.

C. Mathematical Analysis' Outline of the Broadcast Disks Method

In [8] the notion of a disk's workload W_i was introduced as the aggregate client waiting time for the portion of the client queries that refer to pages belonging to the i^{th} disk. More specifically, W_i was defined as:

$$W_i = N_i \cdot D_q \tag{1}$$

where N_i stands for the number of client queries that correspond to pages from the i^{th} disk out of a total number of N queries and D_q denotes the estimate waiting time for a single client query of this kind. Furthermore, it has been proven [8] that (1) can be rewritten as follows:

$$W_i = \left[N \cdot \sum_{l=d_{i-1}}^{d_i} P_p(l) \right] \cdot \frac{L}{2U_i} \tag{2}$$

where d_i, d_{i-1} are the page indices that define the i^{th} disk and P_p is the p.d.f. of the client queries as depicted in Fig. 1. L stands for the size of the broadcast schedule's period measured in number of pages broadcasted and U_i indicates the angular velocity of the i^{th} disk.

In addition, by setting $G(x) = \sum_{l=1}^{x} P_p(l)$, (2) is transformed as follows:

$$W_i = \frac{NL}{2} \cdot \frac{1}{U_i} \left[G(d_i) - G(d_{i-1}) \right], \quad i = 1, \ldots, NoD \tag{3}$$

where NoD represents the total number of disks.

In the context of the same work [8], the uniform workload distribution was examined. In a more general approach, an arbitrary workload distribution can be expressed as:

$$p_1 \cdot W_1 = p_2 \cdot W_2 = \ldots = p_{NoD} \cdot W_{NoD}, \quad p_i \in (0, \infty)$$

which through (3) becomes:

$$\frac{p_1}{U_1} \cdot \left[G(d_1) - 0 \right] = \frac{p_2}{U_2} \cdot \left[G(d_2) - G(d_1) \right] = \ldots$$
$$\ldots = \frac{p_{NoD}}{U_{NoD}} \cdot \left[1 - G(d_{NoD-1}) \right] \tag{4}$$

since $G(d_0) = \sum_{l=1}^{0} P_p(l) = 0$ and $G(d_{NoD}) = \sum_{l=1}^{dbsize} P_p(l) = 1$. Equation 4 can be expressed in matrix form $A \cdot X = B$ as presented in (5).

Thus, $G(d_i), i = 1, \ldots, NoD$, and therefore - since $G(x)$ is 1-1 as a strictly rising function - the d_i points themselves can be generally expressed as:

$$\begin{cases} d_i = \mathcal{F}_i(NoD, \{p_j, U_j, j=1, \ldots, NoD\}), \ i=1, \ldots, NoD \\ d_{NoD} = dbsize \end{cases} \tag{6}$$

where $\mathcal{F}_i$ denotes the relation between the NoD, p_j, U_j parameters and d_i resulting from the solution of the linear system (5).

The parameters p_j and U_j are usually assumed to be terms of uniparametrical numerical series as follows:

$$p_j = \Pi(j, \phi)$$
$$U_j = \Upsilon(j, \Delta)$$

where ϕ, Δ are arbitrarily chosen parameters.

$$\begin{pmatrix} \frac{p_1}{U_1}+\frac{p_2}{U_2} & -\frac{p_2}{U_2} & 0 & 0 & \dots & 0 & 0 & 0 \\ -\frac{p_2}{U_2} & \frac{p_2}{U_2}+\frac{p_3}{U_3} & -\frac{p_3}{U_3} & 0 & \dots & 0 & 0 & 0 \\ 0 & -\frac{p_3}{U_3} & \frac{p_3}{U_3}+\frac{p_4}{U_4} & -\frac{p_4}{U_4} & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 0 & -\frac{p_{NoD-1}}{U_{NoD-1}} & \frac{p_{NoD-1}}{U_{NoD-1}}+\frac{p_{NoD}}{U_{NoD}} \end{pmatrix} \cdot \begin{pmatrix} G(d_1) \\ G(d_2) \\ G(d_3) \\ \vdots \\ G(d_{NoD-1}) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ \frac{p_{NoD}}{UNoD} \end{pmatrix} \quad (5)$$

Thus, (6) can be rewritten as:

$$\begin{cases} d_i = \mathcal{F}_i(NoD, \Delta, \phi), & i = 1, \dots, NoD \\ d_{NoD} = dbsize \end{cases} \quad (7)$$

Finally, an estimation of the mean delay time was given in [8] as:

$$\overline{D} = \frac{\sum_{i=1}^{NoD} W_i}{N} \overset{(3)}{=} \frac{L}{2} \sum_{i=1}^{NoD} \frac{G(d_i) - G(d_{i-1})}{U_i} = \overset{(7)}{=} \mathcal{F}(NoD, \Delta, \phi) \quad (8)$$

Thus, a performance optimization scheme can be established as follows:

1) Define valid value sets $S_{NoD}, S_\Delta, S_\phi$ for the NoD, Δ and ϕ parameters.
2) For every combination $S_{NoD} \times S_\Delta \times S_\phi$:
   i. Solve the linear system (5) and calculate the d_i points in order to define the corresponding disk sizes.
   ii. Proceed to the calculation of the corresponding estimated mean delay time through (8).
   iii. Keep the combination $S_{NoD} \times S_\Delta \times S_\phi$ and disks sizes that achieve the minimum $\overline{D}$.

III. The Proposed Cost-Effective Scheme

As in any typical centralized system, the bulk of the workload in a push system is assigned to the central broadcast server. It is therefore essential to take any steps possible to reduce its hardware needs. Thus, the proposed analysis-based cost-effective scheme consists of two algorithms. The FastOptimization algorithm reduces the computational requirements of the scheme, while the MemoryConserving algorithm promotes minimal memory usage.

A. The FastOptimization algorithm

The issue of enhancing the computational speed can be addressed by breaking down the procedure of II-C into the following four tasks:

1) Calculate the elements of the parameter matrix 5 according to the chosen $S_{NoD} \times S_\Delta \times S_\phi$ combination.
2) Solve the linear system (5).
3) Reverse the $G(d_i)$ values and obtain the d_i points.
4) Find the minimum estimated delay time $\overline{D}$.

The tasks $1, 3$ and 4 are trivial, and standard, computationally optimal corresponding procedures already exist. Due to the peculiar form of the linear system (5) though, task 2, which makes up for the bulk of the computational power required by the procedure, can be significantly accelerated.

The best solution so far was the Thomas algorithm [13] designed for solving any tridiagonal linear $n \times n$ system in $O(n)$ operations instead of $O(n^3)$ required by Gaussian elimination. The method consists of three steps:

Step 1. Firstly, calculate the a_i, b_i, c_i, d_i parameters and form the tridiagonal system in the following manner:

$$\begin{pmatrix} b_1 & c_1 & 0 & \dots & 0 & 0 \\ a_2 & b_2 & c_2 & \dots & 0 & 0 \\ 0 & a_3 & b_3 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & a_n & b_n \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_n \end{pmatrix}$$

Step 2. Then, modify the coefficients as follows:

$$c'_i = \begin{cases} \frac{c_i}{b_i}, & i = 1 \\ \frac{c_i}{b_i - c'_{i-1} \cdot a_i}, & i = 2, 3, \dots, n-1 \end{cases}$$

$$d'_i = \begin{cases} \frac{d_i}{b_i}, & i = 1 \\ \frac{d_i - d'_{i-1} \cdot a_i}{b_i - c'_{i-1} \cdot a_i}, & i = 2, 3, \dots, n \end{cases}$$

Step 3. Finally, obtain the solution by back substitution:

$$x_n = d'_n$$

$$x_i = d'_i - c'_i \cdot x_{i+1}, \quad i = n-1, n-2, \dots, 1$$

In the case of system (5):

- Step 1 requires $(n+1) + n + n - 1 = 3n$ operations, since the matrix is symmetric and $b_i = |c_{i-1}| + |a_i|$, $i = 2, \dots, n$.
- Step 2 requires $1 + 3(n-1) = 3n - 2$ operations, since $d_i = d'_i = 0, i = 1, \dots, n-1$.
- Step 3 requires $2(n-1) = 2n - 2$ operations.

Thus, the Thomas algorithm requires a total of $T_1 = 8n - 4$ operations.

The proposed FastOptimization algorithm is as follows: given that $p_1 \cdot W_1 = p_2 \cdot W_2 = \dots = p_{NoD} \cdot W_{NoD}$, we define $A = p_i \cdot W_i$. Then, because of (3), $A = \frac{LN}{2} \cdot \frac{p_i}{U_i} \cdot \left[G(d_i) - G(d_{i-1}) \right]$. Let $\widetilde{A} = \frac{A}{\frac{LN}{2}} = \frac{p_i}{U_i} \cdot \left[G(d_i) - G(d_{i-1}) \right]$. It holds that:

$$G(d_i) = \frac{U_i}{p_i} \cdot \widetilde{A} + G(d_{i-1}), \quad i = 1, \dots, NoD \quad (9)$$

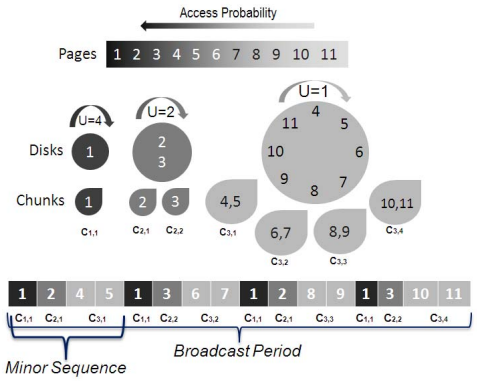


Figure 2. The broadcast schedule construction procedure.

which can be rewritten as:

$$G(d_i) = \frac{U_i}{p_i}\widetilde{A} + \frac{U_{i-1}}{p_{i-1}}\widetilde{A} + \ldots + \frac{U_1}{p_1}\widetilde{A} = \widetilde{A}\sum_{j=1}^{i}\frac{U_j}{p_j}$$

Setting $i = NoD$, the above equation becomes:

$$G(d_{NoD}) = \widetilde{A}\sum_{j=1}^{NoD}\frac{U_j}{p_j}$$

and since $G(d_{NoD}) = 1$:

$$\widetilde{A} = \frac{1}{\sum_{j=1}^{NoD}\frac{U_j}{p_j}} \tag{10}$$

Finally, though (9):

$$G(d_{i-1}) = G(d_i) - \frac{U_j}{p_j}\widetilde{A}, \quad i = NoD, \ldots, 2 \tag{11}$$

and thus the solution is obtained by back substitution.

Equation 10 requires $2NoD$ operations, while (11) requires $2(NoD - 1)$, since the $\frac{U_i}{p_i}$ parameters have been calculated in (10). Thus, the FASTOPTIMIZATION algorithm requires a total of $T_2 = 4NoD - 2$ operations. The size of the parameter matrix n and the number of disks NoD are connected though the relation: $n = NoD - 1$. Thus, $T_2 = 4n + 2$ and $T_1 > T_2$ for every $n > 1$, since $n \in N^+$. Furthermore, for $n \gg 1$, $\frac{T_2}{T_1} \simeq \frac{4n}{8n} = 50\%$.

It is evident that the FASTOPTIMIZATION algorithm requires just the half of the computational operations compared to the Thomas algorithm. Moreover, the simplicity of the new algorithm's implementation means it easier to achieve its full theoretical potential.

B. *The* MEMORYCONSERVING *algorithm*

The standard broadcast schedule constructor algorithm of the Broadcast Disks method [4] states that each disk is to be split in smaller chunks, which then are to be broadcasted in a round-robbin manner as shown in Fig. 2. A brief description is given in Algorithm 1.

Algorithm 1 The schedule constructor algorithm.

Input: The velocities U_i of the NoD disks.
Output: The broadcast program.

1: Calculate the max_chunks as the Least Common Multiple of the disks' velocities U_i, where $i = 1, \ldots, NoD$
2: Split each disk into $num_chunks(i) = \frac{max_chunks}{U_i}$ chunks
3: Broadcast the C_{ij} chunks in a round robin manner.

Algorithm 1 defines the broadcasting of pages' chunks. Yet, for a realistic implementation, it is advantageous to know which single page is to be broadcasted at any given moment. The proposed approach:

1) facilitates the creation of system's simulators as it obviously offers reduced complexity.
2) it promotes the efficiency of client-side traffic prediction scheme.

A relatively simple way to implement a procedure that accepts the current time slot index as input and produces the page to be broadcasted as output is the following:

1) use Algorithm 1 to construct a whole period of broadcast program and store it in a cache memory section
2) at any given time slot T, broadcast the page with index $i = mod(t, L)$, where L indicates the size of the broadcast schedule and as such:

$$L = \sum_{i=1}^{NoD}\lceil\frac{d_i - d_{i-1}}{num_chunks(i)}\rceil \cdot num_chunks(i) \cdot U_i \tag{12}$$

where $d_0 = 0$, $num_chunks(i) = \frac{lcm(U_i, i=1,\ldots,NoD)}{U_i}$ and lcm stands for the least common multiple.

This approach, though simple in conception, has two performance-degrading disadvantages:

1) L increases as the number of diks and their corresponding velocities increase, as depicted in Fig. 3, and therefore so does the size of the required cache memory size.
2) Constructing and storing an enormous broadcast schedule to cache memory adds a considerable overhead to the system's performance.

A more memory-conserving approach would calculate the index of the page to be broadcasted directly from the current time slot without storing any parts of he broadcast schedule to memory. The proposed MEMORYCONSERVING algorithm is presented as Algorithm 2.

The page with index *ALT* is broadcasted when p_B is out of bounds, as mentioned in [4]. It must be stated that constants in lines 1 and 2 of Algorithm 2 need only be calculated once, and can even be skipped and replaced with inline calculations where required. Thus it is obvious that the amount of memory required by this algorithm is minimal.

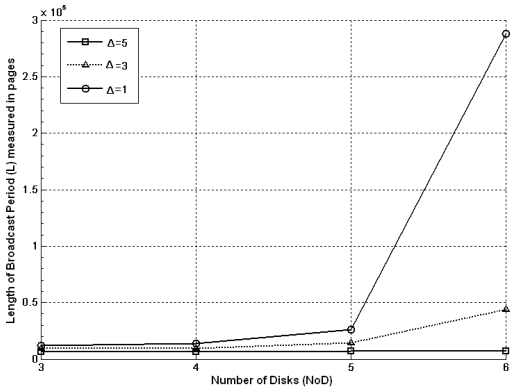


Figure 3. The length of broadcast schedule L as a function of the number of disks NoD. The disks' velocities U_i are set as terms of an arithmetic progression with common difference Δ, i.e. $U_i = (NoD - i) \cdot \Delta + 1$, where $i = 1, \ldots, NoD$.

Algorithm 2 The MEMORYCONSERVING algorithm.

Input: The current timeslot T, the sizes S_i and the velocities U_i of the NoD disks, an alternative value *ALT*.

Output: The page to be broadcasted, p_B.

1: Fill the $chunk_size_i = \lceil \frac{S_i \cdot U_i}{max_chunks} \rceil$ and $num_chunks_i = \frac{max_chunks}{U_i}$ arrays, $i = 1, \ldots, NoD$.
2: Calculate the length of the Minor Sequence $M = \sum_{i=1}^{NoD} chunk_size_i$ and the length of the Broadcast Schedule $L = M \cdot max_chunks$.
3: Calculate the current page index $p = \mod(T, L)$
4: **if** $(p == 0)$ **then**
5: $p = L$
6: **end if**
7: Calculate the page index m in Minor Sequence as $m = \mod(p, M)$
8: set $disk = 0$
9: **while** $m > 0$ **do**
10: $disk = disk + 1$
11: $m = m - chunk_size_{disk}$
12: **end while**
13: $m = m + chunk_size_{disk}$
14: Calculate the index $minor_sequence = \lceil \frac{p}{M} \rceil$
15: Calculate the index $chunk = \mod(minor_sequence, num_chunks_{disk})$
16: **if** $(chunk == 0)$ **then**
17: $chunk = num_chunks_{disk}$
18: **end if**
19: Calculate $low_limit = \sum_{i=1}^{disk-1} S_i$ and $up_limit = \sum_{i=1}^{disk} S_i$
20: Broadcast page $p_B = low_limit + (chunk - 1) chunk_size_{disk} + m$
21: **if** $(p_B > up_limit)$ **then**
22: $p_B = ALT$
23: **end if**

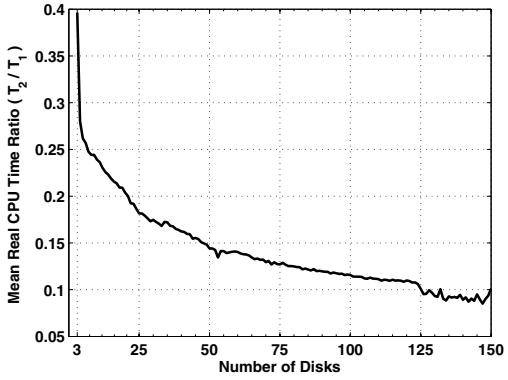


Figure 4. Thomas Method (T_1) vs the FASTOPTIMIZATION algorithm (T_2).

IV. COST EVALUATION

In order to obtain a realistic measure of the cost-efficiency of the presented algorithms, they were implemented in MATLAB alongside their corresponding rivals and were incorporated in a full simulator of the broadcast disks model. The MATLAB profiler was then employed to measure the CPU time spent by each algorithm, when assigned the same task, as well as the required memory. Notice that in terms of client waiting time all algorithms perform the same [8]. The new, cost-related results, along with more detailed descriptions are given in the following subsections.

A. Thomas Method vs the FASTOPTIMIZATION algorithm

Two single functions were implemented, both receiving the arrays $(p_i, U_i, i = 1, \ldots, NoD)$ as input, and producing the corresponding $G(d_i), i = 1, \ldots, NoD$ as output. The first function - T_1 - was based on the Thomas method, while T_2 was the implementation of the FASTOPTIMIZATION algorithm. The NoD parameter was set to vary in the range $[3, 150]$. For each single value of NoD, the CPU time spent in T_1 and T_2 was recorded by means of the MATLAB profiler. The procedure was repeated 100 times in order to obtain a more reliable estimate of the corresponding times. Then, their mean values were calculated and their ratio $\frac{\overline{T_2}}{\overline{T_1}}$ was recorded and displayed in Fig. 4.

Even though it is obvious that the FASTOPTIMIZATION algorithm outperforms the Thomas method in every single case, the ratio $\frac{\overline{T_2}}{\overline{T_1}}$ is much lower than its theoretical values. When calculating an algorithm's complexity, as in Section III-A, the memory allocation, memory-to-CPU-registers and CPU-registers-to-memory transfer commands are not taken into account, as they are subjected to the CPU architecture. Yet these operations also take up a considerable portion of the total CPU time. In addition, the Thomas method demands the allocation and manipulation of four separate variable arrays, namely a_i, b_i, c_i, x_i. On the other hand, the FASTOPTIMIZATION algorithm requires only one array, the

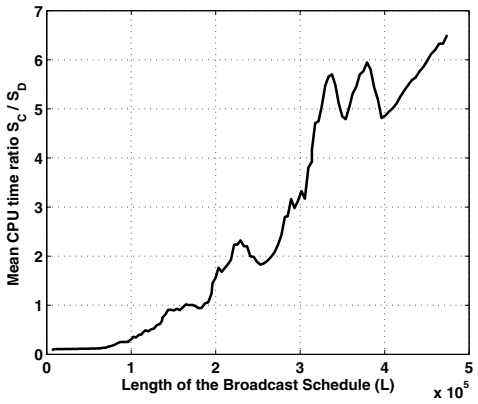


Figure 5. Caching strategy (S_C) vs Broadcasting on demand (S_D).

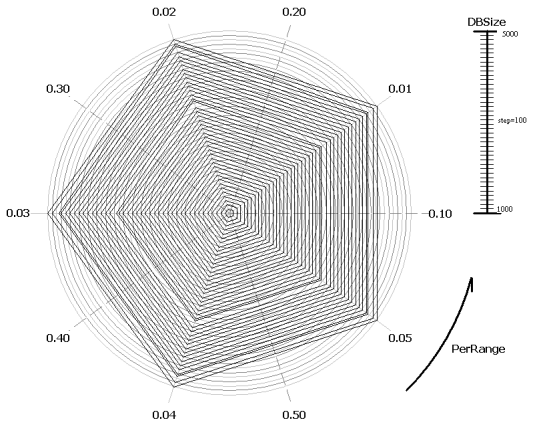


Figure 6. Illustration of the behavior of the NoD parameter, for $\frac{range}{dbsize} = 0.3$ and $\theta = 0.9$ while $dbsize \in [1000, 5000]$ and $\frac{regionsize}{range} \in [0.01, 0.05]$

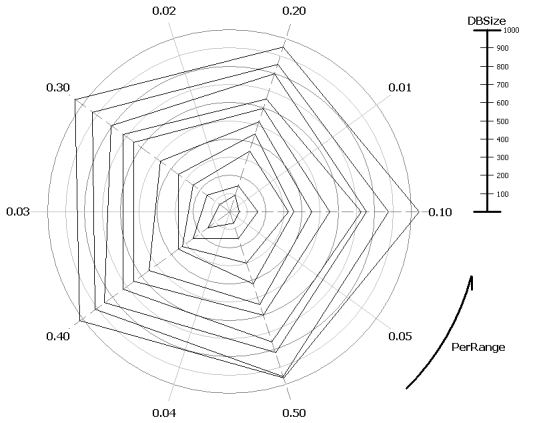


Figure 7. Illustration of the behavior of the Δ parameter, for $\frac{range}{dbsize} = 0.3$ and $\theta = 0.9$ while $dbsize \in [100, 1000]$ and $\frac{regionsize}{range} \in [0.1, 0.5]$

x_i, just to hold the results. Thus the memory allocation and access command are bound to be much less in its case. Finally, the CPU time of a function is implementation-dependent. Better implementations of the Thomas method may be possible than the one used in the context of this paper. Yet this brings to light another advantage of the new algorithm: its simplicity and easy of implementation make it easy to achieve its full theoretical potential, in contrast with the Thomas method which requires programmatic optimization as well.

B. Broadcast schedule caching vs Broadcasting on demand

Again, both strategies described in Section III-B were implemented and incorporated into a full simulator of the broadcast system. The two simulators were identical in every other aspect. The MATLAB profiler was employed once again to calculate the total CPU time spent by each simulator when assigned the same task. The ratio $\frac{\overline{S_C}}{\overline{S_D}}$ was recorded, where $\overline{S_C}$ denotes the mean CPU time spent in a simulation using the caching strategy (Algorithm 1), while $\overline{S_D}$ stands for the broadcasting-on-demand strategy employing simulation (Algorithm 2). A three-disks configuration was used in every case, with sizes $[400, 600, 4000]$. Their velocities were set in an arithmetic progression fashion, $U_i = 1+(NoD-i)\cdot\Delta$, and the common difference Δ ranged accordingly in order to achieve broadcast schedule lengths equally distributed in the range $[0.1, 5] \cdot 10^5$. The results are depicted in Fig. 5. As expected, the caching strategy is superior for small broadcast schedules, performing $10 - 7$ times than its alternative. Both strategies perform equally for $L \in [1.5, 1.8] \cdot 10^5$. For greater lengths, the broadcasting-on-demand strategy is undoubtedly the best choice, performing up to 6.5 times better than the caching strategy.

From a memory usage point of view, the caching strategy requires $U_M = (L) \cdot (sizeofsinglepage)$ memory size, where L stands for the length of the broadcast schedule. As the x-axis of Fig. 5 denotes, $L \in [5000(dbsize), 4.7 \cdot 10^5]$. For a standard page size of $10KB$ (a typical, static HTML page), $U_M \in [50MB, 4.7GB]$. The broadcasting on demand strategy on the other hand requires a steady $U_M^{'} = (1) \cdot (sizeofsinglepage) = 10KB$ amount of memory. It is readily obvious that the caching strategy may pose a bottleneck and even result into an unrealizable system.

V. On the Linearity of the System's Behavior

The performance related observations of the previous sections had a direct impact on the system's simulation time as well, enabling the study of numerous client cases in a relatively small amount of time. For the first time every single one of the $S = \{dbsize, range, regionsize, \theta\}$ set of system defining parameters varied in a predefined range. Purpose was the study of the linearity or non-linearity of the behavior of the optimal values of the $P = \{NoD, \Delta, \phi\}$

performance defining parameters. This revealed another problem: a way of 5-dimensional representation (the four parameters of the S set and one from the P set) had to be improvised. The result is depicted in Fig. 6 and 7.

Each such diagram refers to a different $\{\theta, \frac{range}{dbsize}\}$ combination and illustrates the behavior of one of the parameters in the P set. In each diagram, every concentric circle corresponds to a different $dbsize$ value. Every radial line represents a different value of the $\frac{regionsize}{range}$ ratio. Each intersection of concentric circle-radial line represents the mean value $\overline{p}$ of the parameter in discussion over all the different values of the $\frac{regionsize}{range}$ ratio. By moving from this point along the radial line in a centrifugal fashion, values greater than $\overline{p}$ are represented. Likewise, by moving in a centripetal fashion values smaller than $\overline{p}$ are represented. The point alongside a radial line, which is equidistant from the current circle and the next, represents the maximum value of the parameter in discussion, over all different values of the $\frac{regionsize}{range}$ ratio. Correspondingly, its symmetrical point represents the minimum value of the parameter. Lines interconnect all points, thus forming the pentagons shown in Fig. 6 and 7.

Subsequently, concentric and resemblant pentagons like the ones presented in Fig. 6 yield linearity in the behavior of the parameter in discussion. In contrast, abnormalities like the ones presented in Fig. 7 yield non-linear behavior. These two diagrams, though indicative - as space restrictions hindered the presentation of the totality of the results - present the general conclusion flawlessly: for small values of the $\frac{regionsize}{range}$ ratio, linearity characterizes the behavior of all parameters of the P set. For higher values though, the client's p.d.f. greatly changes in form, and thus non-linearity makes its appearance. The captions of Fig. 6 and 7 present the exact corresponding parameter ranges.

VI. Conclusion

Research on the field of broadcast systems has traditionally overlooked the cost parameter when proposing new, performance related algorithms. However, the primary advantage of these systems is exactly their low implementation cost. In this notion, this paper presented a cost efficient version of the broadcast scheduling procedure of [8] in terms of CPU and memory usage. It was shown that small changes in the computational part procedure can lead up to a 90% reduction in the required CPU time. It was also pointed out that by simply choosing appropriately between caching and calculating-on-demand strategies, the system can benefit up to 7 times in terms of CPU time. From a memory usage point of view, this simple choice not only curtails minimal related needs, but can also be the factor that determines the realizability of a broadcast disks-based push system.

These cost-related observations were readily exploited in order to run thousands of whole system simulations, ranging every single one of the system's configuration parameters, in a small amount of time. Linearity between the achieved client mean waiting time and the parameters' values was observed when the clients access only a small portion (i.e $< 5\%$) of the available data items.

References

[1] A. Bar-Noy, R. Bhatia, J. Naor, and B. Schieberaruch, "Minimizing service and operation costs of periodic scheduling," in *Proc. of SODA'98*.

[2] V. Gondhalekar, R. Jain, and J. Werth, "Efficient access to personalized informationservices via periodic wireless data broadcast," in *Proc. of ICC'97*.

[3] C. Kenyon and N. Schabanel, "The data broadcast problem with non-uniform transmission times," in *Proc. of SODA'99*.

[4] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik, "Data management for asymmetric communication environments," in *Proc. of SIGMOD'95*.

[5] W. Yee, S. Navathe, E. Omiecinski, and C. Jermaine, "Bridging the gap between response time and energy-efficiency in broadcast schedule design," in *Proc. of EDBT'02*.

[6] E. Pitoura and R. Chrusanthis, "Multiversion data broadcast," *IEEE Transactions on Computers*, vol. 51, no. 10, pp. 1224–1230, 2002.

[7] H. Leung, J. Yuen, K. Lam, and E. Chan, "Enhanced multiversion data broadcast scheme for time-constrained mobile computing systems," in *Proc. of DEXA'02*.

[8] C. Liaskos, S. Petridou, G. Papadimitriou, P. Nicopolitidis, and A. Pomportsis, "An analytical approach to the design of wireless broadcast disks systems," in *Proc. of ISADS'09*.

[9] S. Acharya, M. Franklin, and S. Zdonik, "Dissemination-based data delivery using broadcast disks," *IEEE Wireless Communications*, vol. 2, no. 6, pp. 50–60, 1995.

[10] P. Nicopolitidis, G. Papadimitriou, and A. Pomportsis, "Exploiting locality of demand to improve the performance of wireless data broadcasting," *IEEE Transactions on Vehicular Technology*, vol. 55, no. 4, pp. 1347 – 1361, 2006.

[11] P. Nicopolitidis, G. Papadimitriou, and A. Pomportsis, "Multiple-antenna data broadcasting for environments with locality of demand," *IEEE Transactions on Vehicular Technology*, vol. 56, no. 5, pp. 2807 – 2816, 2007.

[12] C. Liaskos, S. Petridou, G. Papadimitriou, P. Nicopolitidis, M. Obaidat, and A. Pomportsis, "Clustering-driven wireless data broadcasting," *IEEE Wireless Communications Magazine*, to appear.

[13] S. Conte and C. DeBoor, *Elementary Numerical Analysis: An Algorithmic Approach*. McGraw-Hill Higher Education, 1980.