An analytical approach to the design of wireless broadcast disks systems

C. K. Liaskos, S. G. Petridou, Member, IEEE, G. I. Papadimitriou, Senior Member, IEEE, P. Nicopolitis, Member, IEEE, A. S. Pomportsis

Department of Informatics
Aristotle University
Thessaloniki, Greece
{cliaskos, spetrido, gp, petros, apombo}@csd.auth.gr

Abstract—The Broadcast Disks method is commonly used to schedule the data transmission in wireless push based networks. The performance of a system that utilizes this method depends upon several parameters. This paper presents the Optimization-Based Procedure (OBP), an analytical approach for estimating the optimal values of these parameters. The validity of the analysis is verified through simulation and comparison with other popular and traditional approaches. The new approach was found to be accurate and dominant in the vast majority of a wide set of test cases. It also provided a thorough insight of the system’s stability and behavior in general.

Keywords- wireless push system; broadcast disks; analytical approach; mean response time

I. INTRODUCTION

Research and evaluation of the performance of wireless communication systems has traditionally relied on computer simulations over the past years. Analytical approaches on the other hand, are very scarce. Simulations do offer advantages as important as speed, ease and flexibility in the process of analysis, but tend to provide obscure insight of a system’s internal way of operation and poor explanation of its behavior overall. The goal of the present work is to present the Optimization-Based Procedure, which is an analytical optimization of the performance of push-oriented wireless communication systems that form their broadcast schedule according to the Broadcast Disks method [1].

The Broadcast Disks method can be briefly described as a lax broadcast scheduling framework that ensures the periodicity and proportionality of the transmission schedule of a push-based system: the interval between two consecutive appearances of the same data item in the schedule is held constant and the total number of its appearances proportional to its popularity. The aforementioned characteristics imply the existence of a specific system architecture which is depicted in Fig. 1. It comprises of a database containing a fixed number of data items eligible for broadcasting, a central broadcasting server responsible for maintaining the transmission schedule’s accordance with the clients’ demands, and finally the clients themselves who provide a metric of their needs to the server by means of a feedback mechanism. According to the Broadcast Disks method, the data items are organized in line with their popularity into a predefined number of groups called disks. These abstractedly represent an array of physical disks spinning around a common axis. Each of these disks is set to rotate with an angular velocity proportional to the aggregate popularity of its contained pages. Then, an imaginary set of stationary heads retrieves pages from the disks and forwards them to the broadcasting system in the same order that they have been read. Thus the broadcast’s periodicity and proportionality criteria are met.

The mathematical analysis that took place focused on the definition of the optimal grouping of the data items into disks, as well as the calculation of their corresponding optimal angular velocities. In all cases we consider as “optimal” the choice that minimizes the server’s mean response time. It must also be stated that the optimization process relied on several assumptions, due to the fact that the complexity of the Broadcast Disks method renders a more straightforward approach extremely difficult. These assumptions generally concern the desirable workload of the disks and the grouping of pages in a scale of equiprobable teams rather than the traditional straightforward grouping at single page level. In addition to performance optimization issues, the analysis also led to an explanation of the system’s seemingly volatile behavior.

In the context of this paper, simulation was used only as a means of checking the validity of the analysis results by comparison with other well-known page grouping and disk spinning schemes. To this end, the GREEDY [2] grouping algorithm was employed. Concerning the corresponding disk spinning scheme, a large number of simulations were executed for each case and the disk velocities achieving the best response time were held. Such a technique can characterized as “unfair” since under realistic conditions each scenario can only be executed once. Still, the analytical approach yielded not only better performance in the vast majority of the test cases, but much smaller processing times as well.

The remainder of this paper is organized as follows: Section II provides the required theoretical background: the network’s architecture, operation and probabilistic...
characteristics are discussed in greater depth. Essential elements of the Broadcast Disks method are presented as well. The mathematical analysis follows in Section III. Simulation configuration and results are presented in Section IV. Finally, conclusions and future work insights are given in Section V.

II. THEORETICAL BACKGROUND

A. Network Architecture

The physical part of the network under discussion typically consists of a server, a database, one or more clients and an asymmetric communication channel. The database containing a number of DBsize data items-most commonly referred to as pages-is connected to the server. All pages are considered to be of equal size. The server is properly equipped (hardware and protocol wise) in order to act as a central node of a cellular network. Any adequately equipped portable device can then act as a client.

The communication channel is asymmetric, either because of the electromagnetic properties of the transmission medium or as a consequence of hardware limitations. From a client’s point of view this stands for adequate download bandwidth, and a very limited—but not zero-upload capability. The assumption that the clients are capable of transmitting data as well is a point of differentiation from bibliography [1], [2], where the upload bandwidth is set to null. Yet this condition does not upset the push-based nature of the system. The data transmission is not only literally minimal, but also absolutely necessary for the implementation of any kind of feedback system, as will become evident in the next subsection. Moreover, all similar papers actually rely on such a feedback mechanism—and therefore on the above condition as well—but do not supply any adequate specifications for it [1]-[5]. In addition, the system does not rely on the transmission ability of the client for its function, but rather for the continual improvement of its performance.

Any client is considered to access only a random subset of the server’s DBsize pages, and their number will be defined as Range. These pages are then organized further in equally sized groups called Regions. All RegionSize pages in one Region have equal probability of being requested by the client. The Regions themselves are considered to follow the zipf probability distribution. Thus if all Regions are sorted according to their popularity in a descending fashion, the request probability of any page contained in a Region $r_j$ will be

$$P_h(r_j) = \frac{c}{\theta} \cdot \frac{\text{Range}}{\text{RegionSize}}, \quad j = 1,...,\left\lfloor \frac{\text{Range}}{\text{RegionSize}} \right\rfloor$$

(1)

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

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

B. Network Operation and the Broadcast Disks Method

The network’s function is a periodical procedure, and its description will begin from the point where the broadcast schedule has been constructed and an arbitrary page is being broadcasted by the server. If and only if this page is actually needed by a client, he replies with a single approval message (e.g. one binary digit in a noiseless environment). Thus the server acquires a posteriori knowledge about the clients’ demands, with a minimal client upload rate. This feedback philosophy is perhaps the only one fitted for a push-based system. Anything else would fall into a client request-server response scheme, which is known to be inappropriate for asymmetric communications, as it would eventually require bigger upload bandwidth from the client, and a great deal of request management effort from the server. This feedback scheme is a based on the one presented in [6]. The server maintains a Votes registry, which holds the aggregated votes for each page in the server database. Upon receiving one client’s vote, the server updates this registry by increasing by one unit the broadcasted page’s registry value. This process is repeated for every broadcasted page, for a valid period of voting time. The optimal duration of this voting mode can be estimated if we assume that some statistical properties of the network’s traffic are known. For example, if we have observed that the real $\theta$ parameter of the client’s zipf p.d.f. is approximately equal to a constant value, we can set the voting period to end when the calculated $\theta$ reaches this value. Actual calculation of this interval is not amongst the purposes of this paper though. At the end of the voting period, the Votes registry has by approximation the form shown in Fig. 2.

At this point the Broadcast Disks method intervenes, initially with the use of a grouping algorithm on the completed Votes registry in order to create a predefined number of clusters of pages with similar access probability. As mentioned earlier, these clusters represent the well-known broadcast disks of the aforementioned method. These disks must then be assigned proper angular velocities. The most common approach is to use an arbitrary constant value called $\Delta$ (Delta) and set the angular velocities of the disks in an arithmetic progression fashion. Thus

$$U_i = (\text{NoD} - i) \cdot \Delta + 1$$

(2)

where $\Delta$ is the common difference of the sequence, NoD stands for the predefined number of disks, and $i=\text{NoD}$ obviously denotes the slowest and last of them.
Finally, having defined the disk sizes and the disk velocities, the method proceeds to create the actual broadcast program in the following manner:

1) Calculate a \( \text{max}_\text{chunks} \) parameter as the Least Common Multiple of the disks’ velocities

2) Split each disk into \( \text{chunks} = \text{max}_\text{chunks} / U_i \) chunks.

3) Create the broadcast program as follows:
   - for \( i=0, \ldots, \text{max}_\text{chunks} \)
   - for \( j=1, \ldots, \text{NoD} \)
   - Broadcast chunk \( C_{i,j} \) \( (i \mod \text{num}_\text{chunks}(j)) \)
   - end
   - end

where \( C_{i,j} \) refers to the \( j \)th chunk of the \( i \)th Disk. The division of a disk into an integer number of equally sized chunks is usually not possible without appropriately zero padding the disk first. These zeros can then be substituted with e.g. the most popular of the server’s pages or in any other way one considers appropriate.

III. MATHEMATICAL ANALYSIS

Assume that the cyclically repeated broadcast program has been constructed and that it comprises of \( L \) equally sized transmissions of pages \( p_i \), where \( i \in [1, \text{DBSize}] \) and \( L \gg \text{DBSize} \). If a page originated from a disk with an angular velocity of \( U_i \), it will appear exactly \( U_i \) times inside the schedule, provided that the speed of the last and slowest disk is equal to one unit. Uniform distribution of all same page appearances inside the schedule is ensured through the use of the Broadcast Disks method, as stated already. Thus the mean waiting time of a client expecting a page \( p_i \) belonging to a disk with velocity \( U_i \) will be

\[
D(p_i) = \frac{L}{2 \cdot U_i} \tag{3}
\]

Additionally, it is assumed that enough time has elapsed for the system to acquire an adequate approximation of the Range, RegionSize and \( \theta \) (zipf p.d.f.) parameters through the feedback mechanism described earlier. Thus the server has sufficient knowledge of the client queries’ p.d.f, which is depicted in Fig. 2 and will be symbolized as \( P_0(p_i) \). \( P_\theta(p_i) \) and \( P_\theta(r) \), Equation (2), are connected with the relation

\[
P_\theta(r_i) = P_\theta(r \cdot \text{RegionSize}) \tag{4}
\]

The oncoming grouping of pages into a number of NoD disks is tantamount to choosing the points denoted as \( d_i \in [1, \text{NoD}] \) in Fig. 2. The first major assumption states that equiprobable pages should not be split over different disks. In other words the \( d_i \) points can only be placed at the end of a Region. As a consequence, the point \( d_{\text{NoD}-1} \) coincides with \( p_{\text{RegionSize}} \) and \( d_{\text{NoD}} \) with \( p_{\text{DBSize}} \).

Now, consider a series of \( N \) queries \( q_i \) of a single client where \( q_i \in [p_1, p_{\text{RegionSize}}] \). Out of these \( N \) queries, exactly

\[
N_j = N \sum_{i=1}^{d_j} P_\theta(p_i) \tag{5}
\]

refer to pages that belong to the first disk. Because of (4) and (1), (5) can be rewritten as

\[
N_j = N \cdot \text{RegionSize} \cdot \sum_{i=1}^{d_j} P_\theta(r_i) = N \cdot \text{RegionSize} \cdot \text{RegionSize} \cdot \frac{1}{\theta} \tag{6}
\]

In a similar manner, for the remaining disks we have

\[
N_j = \text{RegionSize} \cdot \text{NoD} \cdot \sum_{i=1}^{d_j} \frac{1}{\theta} \cdot j = 2, \ldots, \text{NoD} \tag{7}
\]

and obviously \( N_{\text{NoD}}=0 \) since the last disk contains all the unused pages. For the sake of presentation simplification we define

\[
K = \text{RegionSize} \cdot \text{NoD} \cdot \frac{1}{\theta} \tag{9}
\]

and thus (6) and (7) can be rewritten as:

\[
N_j = K \cdot G(d_j) \tag{10}
\]

\[
N_j = K \cdot \sum_{i=1}^{d_j} G(d_i) - G(d_{j-1}) \tag{11}
\]

From (10), (11) and (3) is derived that the aggregate waiting time \( W_j \) for pages belonging to the \( j \)th disk is

\[
W_j = N_j \cdot \frac{L}{2U_j} \tag{12}
\]

\( W_j \) will be referred to as the \( j \)th disk’s workload. The server’s mean response time can then be calculated as

\[
D = \frac{\sum_{j=1}^{\text{NoD}} W_j}{N} \tag{13}
\]

The second major assumption declares that the workloads of all broadcast disks should be equal. Disks with null
workload are obviously excluded from this rule. It must be stated that this assumption may not yield optimal response times, but instead represents the optimal approach for avoiding performance bottlenecks. This assumption can be expressed as

\[ W_1 = W_2 = W_3 = \ldots = W_{\text{NoD}-1} \]

and because of (10), (11) and (12)

\[ \frac{K-L(G(d_1))}{2} \cdot \frac{K-L(G(d_2))}{2} \cdot \frac{K-L(G(d_{\text{NoD}}))}{2} = \frac{K-L(G(d_{\text{NoD}-2}))}{2} \]

which is simplified as follows:

\[ \frac{G(d_1)}{U_1} = \frac{G(d_2) + G(d_1)}{U_2} = \ldots = \frac{G(d_{\text{NoD}-1}) - G(d_{\text{NoD}-2})}{U_{\text{NoD}-1}} \]  \hspace{1cm} (14)

Considering the disk speeds \( U_i \) as simple parameters and the \( G(d_i') \) as unknown variables, (14) represents a linear \((\text{NoD}-2)\times(\text{NoD}-2)\) system which can be easily expressed in matrix form \((Ax=B)\):

\[
\begin{bmatrix}
\frac{1}{U_1} & \frac{1}{U_2} & 0 & 0 & \ldots & 0 & 0 & 0 \\
\frac{1}{U_3} & \frac{1}{U_2} & \frac{1}{U_2} & 0 & \ldots & 0 & 0 & 0 \\
0 & \frac{1}{U_2} & \frac{1}{U_2} & \frac{1}{U_2} & \ldots & 0 & 0 & 0 \\
0 & 0 & \frac{1}{U_2} & \frac{1}{U_2} & \frac{1}{U_2} & \ldots & 0 & 0 \\
0 & 0 & 0 & \frac{1}{U_2} & \frac{1}{U_2} & \frac{1}{U_2} & \ldots & 0 \\
0 & 0 & 0 & 0 & \frac{1}{U_2} & \frac{1}{U_2} & \frac{1}{U_2} & \ldots \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
\frac{1}{U_{\text{NoD}-1}} & \frac{1}{U_{\text{NoD}-2}} & \frac{1}{U_{\text{NoD}-3}} & \frac{1}{U_{\text{NoD}-4}} & \ldots & \frac{1}{U_{\text{NoD}-2}} & \frac{1}{U_{\text{NoD}-1}} & 1
\end{bmatrix}
\begin{bmatrix}
G(d_1) \\
G(d_2) \\
G(d_3) \\
G(d_4) \\
\vdots \\
G(d_{\text{NoD}-2})
\end{bmatrix}
= \begin{bmatrix}
\frac{1}{U_1} \\
\frac{1}{U_2} \\
\frac{1}{U_2} \\
\frac{1}{U_2} \\
\vdots \\
\frac{1}{U_{\text{NoD}-1}}
\end{bmatrix}
\]

\hspace{1cm} (15)

The parameter matrix \( A \) of this linear system is a symmetric tridiagonal one. Furthermore it satisfies the criteria

\[ a_{i,i} \geq a_{i,i+1}, \quad i=2, \ldots, n-1, \]

Thus the system always has a unique solution [7]. In addition, the convergence to this solution is known to be very quick, requiring \( O(\text{NoD}) \) operations [7], with the optimal \( \text{NoD} \) parameter itself typically ranging from 1 to 10. Finally it must be stated that since the \( U_i \) parameters all derive from Equation (2), the only parameter actually requiring definition is the common difference \( \Delta \).

At this point, the server’s response time optimization procedure can be fully defined as a series of simple steps described below:

1. A value set is defined for the \( \Delta \) parameter. Typical choices are the ranges \([1,100]\) or \([1,150]\).
2. In a similar manner, a range of values is selected for the \( \text{NoD} \) parameter, typically \([2, 10]\) to \([2,15]\).
3. In each of the above cases much larger value sets may be used with trivial processing time impact. Actual results though have indicated the use of values beyond the aforementioned ranges to be excessive.
4. For every possible pair of values \( (\Delta, \text{NoD}) \) the disk speeds \( U_i \) are calculated through Equation (3), and the linear system of (15) is solved. Thus the \( G(d_i') \), \( i=1, \ldots, \text{NoD} \) are defined. Through Equations (10), (11), (12) and (13) the server’s mean response time is calculated.

The pair \( (\Delta, \text{NoD}) \) achieving the minimum is considered to optimize the server’s performance.

Finally, attention must be paid to the fact that the value of the parameter \( L \) is subjected each time to the current disk speeds and sizes as implied in subsection II-B and thus must be calculated accordingly when needed, as described below:

\[ L = \sum_{i=1}^{\text{NoD}} \frac{d_i - d_{i-1}}{\text{num\_chunks}(i)} \times \text{num\_chunks}(i) \times U_j \]

where \( d_0 = 0 \) and

\[ \text{num\_chunks}(i) = \text{lcm}(U_i; i=1, \ldots, \text{NoD}) / U_i \]

where \( \text{lcm} \) stands for the least common multiple.

IV. SIMULATION

A. Configuration

The goal of the simulation was the estimation of the response times achieved by the aforementioned procedure for every major client case. The term client case refers to any different combination of the Range, RegionSize and \( \theta \) parameters, as it describes a unique client probabilistic model. Amongst them, the Range parameter is subjected to the total amount of available server pages (DBSize). However, since the totality of preceding papers set this parameter to the value 5000, the same convention will be used in the context of this work as well. The value sets of the remaining parameters are presented in Table I.

The values for the Range and RegionSize parameters are chosen in tandem for obvious reasons. Concerning the \( \theta \) parameter, values smaller than one unit correspond to high degree of correlation between regions, while values in the range \((1, \infty)\) suggest the opposite. The values 0.3 and 1.5 represent the two corresponding extremes. The value \( \theta = 1 \) is not allowed due to a zipf p.d.f. restriction. Finally, the value 0.95 is the most commonly used in bibliography [1], [2]. For each client case the procedure of section III produces the

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \theta )</td>
<td>0.3 0.5 0.7 0.95 1.1 1.6</td>
</tr>
<tr>
<td>Range/RegionSize</td>
<td>30/1 30/3 30/5 1000/30 1000/50 2000/50 2000/100 3000/50 4000/50</td>
</tr>
</tbody>
</table>
optimal $\Delta$ and NoD and then the corresponding disk sizes and speeds, exactly as already described.

Moreover, a well known and approved similar procedure is required in order to serve as a measure of comparison, and thus as a means of validating the soundness of the preceding mathematical analysis. This role is assigned to the GREEDY grouping algorithm, a lightweight and very efficient scheme. GREEDY operates by grouping pages rather than regions into disks, and generally relies on the recursive splitting of the fastest disk at one pre-calculated “low cost” point. It must be stated that GREEDY requires the NoD parameter as input and is absolutely irrelevant to the setting of the disks’ velocities. Thus the $\Delta$ parameter has to be otherwise set as well. For the sake of the comparison and in order to overcome this lack of counter proposed $\Delta$ and NoD values, the simulation in the case of GREEDY is executed for a wide set of values of the aforementioned parameters, as presented in Table II. The best response time is held and compared with the one resulting from the optimization procedure of Section III. The scheme used in the case of GREEDY is of course of no realistic use and obviously yields extremely long processing times, but serves the comparison purposes well.

Each simulation begins with the assumption that the Votes Registry is completed and has the perfect form of Fig 2. For an amount of 15,000 client queries the server’s response time is observed and its mean value is calculated. Finally, in order to be in full compliance with [1] where the broadcast disks method was introduced, the client’s ThinkTime is taken into account and is set equal to 2 timeslots. In other words, upon receiving a requested page the client remains inactive for a period of 2 timeslots.

**B. Simulation Results and Remarks**

The results are displayed in Tables III and IV. Each column represents a specific Range/RegionSize pair of values, while the row names denote the corresponding values of the $\theta$ parameter of the client’s zipf p.d.f. Thus each table cell represents a unique client configuration case. The acronyms OBP and GBP are used to differentiate the Optimization-Based Procedure from the GREEDY-Based Procedure respectively. Due to the vast number of the results and in order to better compare OBP and GBP, we present only the smallest achieved mean response time (s.m.r.t.) per procedure, accompanied by the corresponding $\Delta$ and disk configuration for each of these cases.

**TABLE II. VALUE SETS OF THE $\Delta$ AND NoD PARAMETERS (GREEDY ONLY)**

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta$</td>
<td>1 2 3 4 5 6 7 8 15 20 50 80 100</td>
</tr>
<tr>
<td>NoD</td>
<td>2 3 4 5 6 7 8 9 10</td>
</tr>
</tbody>
</table>

Slight variations in the response time are noticed in some cases that share the same $\Delta$ and disk configuration (e.g. cells (1, 1) and (1, 2) of Table III). These values are theoretically expected to be exactly equal, but this assumption is not accurate in our case. The simulation model used to extract the aforementioned results relies on multiple randomly initialized parameters (e.g. the series of the actual client queries) and thus variations of ±15 time units have been observed and should be considered natural.

OBP states that the optimal disk configurations must comprise of exactly three disks rotating at high speeds (as indicated by the high corresponding $\Delta$ values) with no exception. GBP presents no similar patterns. Evidently, OBP outperforms GBP in the vast majority of the test cases. The difference in performance is inversely proportional to the $\theta$ parameter for each Range/RegionSize combination. Both procedures produce similar results for very small Ranges as one could expect, but OBP is clearly dominant in every other case, with two exceptions that respect the last two cells of Table IV ($\theta$=1.5 and Range/RegionSize=3000/50, 4000/50). In order to explain this phenomenon, indicative disk workloads’ where calculated through simulation and are graphically illustrated in Fig. 3. It is evident that OBP correctly creates approximately equally loaded disks, in contrast with GBP which loads them in a non uniform way. Similar results are produced for Range=4000/RegionSize=50 as well. It is thus concluded that the equally loaded disks policy is inappropriate for very large Ranges.

Although the dominance of OBP over GBP may serve as sufficient proof of the analysis’s validity, it is commendable to ensure it even further. In Table 5 the smallest mean response time estimated through the mathematical analysis of Section III is compared with the corresponding simulation results. Both calculations yield full accordance. Minor variations are ascribed to the randomly initialized parameters of the simulation procedure. Table V may present only a small number of indicative cases, yet similar results were derived from the totality of the simulation process.

![Figure 3. Disk Workloads calculated through simulation](image-url)
### TABLE III. SIMULATION RESULTS FOR RANGES 30 AND 1000

<table>
<thead>
<tr>
<th>θ</th>
<th>Range / Region</th>
<th>Size 30/1</th>
<th>30/3</th>
<th>30/5</th>
<th>1000/30</th>
<th>1000/50</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>s.m.r.t</td>
<td>98</td>
<td>101</td>
<td>103</td>
<td>660</td>
<td>696</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>15</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>[Disk Sizes]</td>
<td>[18 12 4970]</td>
<td>[20 10 4970]</td>
<td>[570 430 4000]</td>
<td>[600 400 4000]</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>s.m.r.t</td>
<td>95</td>
<td>88</td>
<td>100</td>
<td>695</td>
<td>639</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>[Disk Sizes]</td>
<td>[12 18 4970]</td>
<td>[13 17 4970]</td>
<td>[386 592 4022]</td>
<td>[398 589 4013]</td>
<td></td>
</tr>
<tr>
<td>0.7</td>
<td>s.m.r.t</td>
<td>93</td>
<td>96</td>
<td>89</td>
<td>706</td>
<td>712</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>6</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>[Disk Sizes]</td>
<td>[10 20 4970]</td>
<td>[11 19 4970]</td>
<td>[331 618 4051]</td>
<td>[347 620 4033]</td>
<td></td>
</tr>
<tr>
<td>0.95</td>
<td>s.m.r.t</td>
<td>98</td>
<td>98</td>
<td>91</td>
<td>839</td>
<td>845</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>[Disk Sizes]</td>
<td>[9 21 4970]</td>
<td>[10 20 4970]</td>
<td>[87 165 546 4257]</td>
<td>[275 602 4123]</td>
<td></td>
</tr>
<tr>
<td>1.1</td>
<td>s.m.r.t</td>
<td>88</td>
<td>83</td>
<td>79</td>
<td>445</td>
<td>447</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>35</td>
<td>35</td>
<td>35</td>
<td>35</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>[Disk Sizes]</td>
<td>[7 23 4970]</td>
<td>[8 22 4970]</td>
<td>[121 790 4000]</td>
<td>[228 592 4190]</td>
<td></td>
</tr>
<tr>
<td>1.5</td>
<td>s.m.r.t</td>
<td>101</td>
<td>95</td>
<td>80</td>
<td>390</td>
<td>410</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>[Disk Sizes]</td>
<td>[3 27 4970]</td>
<td>[6 24 4970]</td>
<td>[90 910 4000]</td>
<td>[141 426 4433]</td>
<td></td>
</tr>
</tbody>
</table>
TABLE IV.  SIMULATION RESULTS FOR RANGES 2000, 3000 AND 4000

<table>
<thead>
<tr>
<th>θ</th>
<th>Range / RegionSize</th>
<th>2000/50</th>
<th>2000/100</th>
<th>3000/50</th>
<th>4000/50</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>OBP</td>
<td>1078</td>
<td>1083</td>
<td>1602</td>
<td>2080</td>
</tr>
<tr>
<td></td>
<td>s.m.r.t</td>
<td>47</td>
<td>44</td>
<td>22</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>[Disk Sizes]</td>
<td>[1150 850 3000]</td>
<td>[1200 800 3000]</td>
<td>[1700 1300 2000]</td>
<td>[2250 1750 1000]</td>
</tr>
<tr>
<td>0.5</td>
<td>s.m.r.t</td>
<td>1867</td>
<td>1870</td>
<td>2200</td>
<td>2378</td>
</tr>
<tr>
<td></td>
<td>Δ</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>[Disk Sizes]</td>
<td>[1390 3610]</td>
<td>[1562 3474]</td>
<td>[1703 3297]</td>
<td>[1858 3142]</td>
</tr>
<tr>
<td>0.7</td>
<td>s.m.r.t</td>
<td>1032</td>
<td>1055</td>
<td>1506</td>
<td>2080</td>
</tr>
<tr>
<td></td>
<td>Δ</td>
<td>38</td>
<td>19</td>
<td>32</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>[Disk Sizes]</td>
<td>[1000 1000 3000]</td>
<td>[1000 1000 3000]</td>
<td>[1450 1550 2000]</td>
<td>[1900 2100 1000]</td>
</tr>
<tr>
<td>1</td>
<td>s.m.r.t</td>
<td>1871</td>
<td>1870</td>
<td>2152</td>
<td>2274</td>
</tr>
<tr>
<td></td>
<td>Δ</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>[Disk Sizes]</td>
<td>[1390 3610]</td>
<td>[1452 3548]</td>
<td>[1507 3493]</td>
<td>[1726 3274]</td>
</tr>
</tbody>
</table>

TABLE V.  ESTIMATED VS SIMULATED RESULTS

<table>
<thead>
<tr>
<th>Range / RegionSize / θ</th>
<th>Estimated s.m.r.t</th>
<th>Simulation results</th>
</tr>
</thead>
<tbody>
<tr>
<td>30 / 1 / 0.3</td>
<td>95</td>
<td>90</td>
</tr>
<tr>
<td>30 / 3 / 0.3</td>
<td>95</td>
<td>93</td>
</tr>
<tr>
<td>30 / 5 / 0.3</td>
<td>95</td>
<td>93</td>
</tr>
<tr>
<td>1000 / 30 / 0.7</td>
<td>510</td>
<td>513</td>
</tr>
<tr>
<td>1000 / 50 / 0.95</td>
<td>498</td>
<td>471</td>
</tr>
<tr>
<td>2000 / 50 / 0.95</td>
<td>886</td>
<td>867</td>
</tr>
<tr>
<td>2000 / 100 / 1.1</td>
<td>876</td>
<td>859</td>
</tr>
<tr>
<td>3000 / 50 / 1.1</td>
<td>1209</td>
<td>1188</td>
</tr>
<tr>
<td>4000 / 50 / 1.5</td>
<td>1440</td>
<td>1408</td>
</tr>
</tbody>
</table>
Finally, the analysis led to important observations on the system’s behavior. Fig. 4 depicts the server’s estimated response time versus the Delta ($\Delta$) parameter—which as stated already represents the disks’ speeds—for a specific client case and for several numbers of disks (NoD). In all cases the system’s performance improves with the speed increase until a certain minimum point has been achieved. From that point and on performance deteriorates in a radical and unstable way. Increasing the number of disks results in reaching the minimum point earlier and also in decreasing the system’s stability.

V. CONCLUSION AND FUTURE WORK

In the context of the present work an analytical optimization of the performance of wireless push-based systems has been presented. The analysis focused on minimizing the mean response time of a server that schedules its broadcast through the use of the Broadcast Disks Method, mainly under a logical assumption concerning the uniform distribution of the disks’ load. The results of the analysis were validated through simulation and comparison with another well-known and efficient similar scheme and were found dominant in the vast majority of the test cases. Several important conclusions concerning the behavior and the stability of such a system were also extracted through the analysis. Future work insights include the benchmarking of other disk loading policies that will not rely on uniformity and may be proven very efficient for serving highly congested push-based networks.

REFERENCES