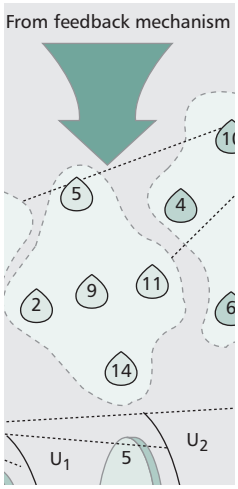
ACCEPTED FROM OPEN CALL

CLUSTERING-DRIVEN WIRELESS DATA BROADCASTING

CHRISTOS K. LIASKOS, SOPHIA G. PETRIDOU, GEORGIOS I. PAPADIMITRIOU, PETROS NICOPOLITIDIS, AND ANDREAS S. POMPORTSIS, ARISTOTLE UNIVERSITY
MOHAMMAD S. OBAIDAT, MONMOUTH UNIVERSITY



The authors introduce and evaluate the Clustering-driven Wireless Data Broadcasting, a complete instantiation of the Broadcast Disks method. The proposed CWDB procedure addresses the major omissions of preceding schemes.

ABSTRACT

The performance of a push-based system relies heavily on the proper scheduling of the broadcast data. To this end, the Broadcast Disks method is most commonly employed. It defines a procedure consisting of four separate algorithms: one to provide and handle the clients' feedback, another to group the data objects into disks, a third one to define their spinning velocities, and finally a Broadcast Sequence constructor algorithm. In this article we introduce and evaluate Clustering-Driven Wireless Data Broadcasting (CWDB), a complete instantiation of the Broadcast Disks method. The proposed CWDB procedure addresses the major omissions of preceding schemes, including the total lack of feedback mechanisms, extremely limited variety of spinning velocity definition algorithms, and no thorough and realistic testing of complete combinations of algorithms under various client configurations. A new efficient clustering-driven data grouping algorithm is also introduced. Extensive simulation results are presented, which indicate that the proposed CWDB is absolutely dominant over other classical methods in the vast majority of the test cases and led to very important conclusions that were previously overlooked.

INTRODUCTION

Push technology, or server push, generally describes a way of communication where the request for a given transaction originates from the publisher or central server: a number of data items are broadcast according to a schedule that best fits the needs of all interested clients. It is contrasted with pull technology where the request for the transmission of information originates from the client and then the server responds accordingly. As the reader perceives, the main factor that affects the performance of push-based systems is the definition of a client set-wise optimal broadcast schedule. This, in turn, requires server-side knowledge of the clients' preferences, which must be provided through a simple (both concept and hardware/software realization-wise) feedback mechanism. This can be a real challenge when designing wireless push-based systems, since simplicity may contradict the efficiency of the feedback scheme, and thus the adaptation factor of the system. Research on the field is orientated mainly toward improvement of both the broadcast scheduling algorithms and hardware implementations of client-side equipment. Regarding the popularity of the opposing push and pull approaches, the former has been progressively gaining more and more of the wireless communications industry's interest over the past few years [1]. The recent spread of technologies that utilize higher frequency bands may have facilitated the evolution of pull-based wireless services and systems, but it has also laid the groundwork for perfection of push-oriented ones. Furthermore, advantages such as lower hardware requirements (client and server) and minimal to zero electromagnetic emission (client) have traditionally favored the use of the latter in cases where economy, battery life, portability, and low electromagnetic interference are prioritized over direct access on demand. Examples traditionally include informative services of hospitals and airports, but these uses are now extended to include generic instant messaging and multimedia broadcasting services over mobile telephony networks and the Internet.

Push-based systems owe much of their aforementioned advantages to their simple 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, as depicted in Fig. 1. The database containing the broadcast data is connected to or is part of the server. This server is also properly equipped (hardware and protocol) to act as a central node of a cellular network. Any adequately equipped portable device can then act as a client.

The functionality of a push-based system typically relies on a server publish/client subscribe logic. The system's central publishing server administers a repeating broadcast program of

1536-1284/09/$25.00 © 2009 IEEE

data objects or *pages* usually organized in *channels* according to their content. The server periodically edits this program by increasing or decreasing the number of occurrences of each page, or adding new ones, in an effort to keep it in line with the clients' demands. A client simply subscribes to one or more channels of his/her choice and reads data from them in a streaming fashion. The clients' preferences may be static and known in advance, but in the most common scenario they must be periodically provided to the server by some means.

The communication channel is assumed to be asymmetric, either because of the electromagnetic properties of the transmission medium or as a consequence of any hardware limitations. From a client's point of view this stands for adequate download bandwidth, and 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 similar literature [2, 3], where the upload bandwidth is set to null. However, it must be stated that this condition does not upset the push-based nature of the system. The required data transmission is not only literally minimal, but also absolutely necessary for the implementation of any kind of automated system providing feedback for the clients' preferences. Moreover, related 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. In addition, the system does not rely on the transmission ability of the client for its basic function, but rather for the continual improvement of its performance.

In order to facilitate the analysis of push-based systems, certain assumptions are usually made:

- Any client is considered to access only a random subset of the server's channels.
- Each channel contains an approximately equal number of pages.
- All pages in each channel have equal probability of being requested by the client.
- Each client is primarily interested in one particular channel of his/her choice, and his/her interest in the rest of them is proportional to their relation with the primary one.

Mathematically, this is most commonly expressed by assuming that the channels access probability follows the Zipf probability distribution function.

The analysis itself traditionally relies on simulation of the network, and the aforementioned assumptions serve the purpose of defining the characteristics of the system's traffic. The actual optimization problem, though, is to define the broadcast program that better suits the needs of the totality of clients. The most common program generation routine generally consists of four separate steps and is referred to as the Broadcast Disks method.

Broadcast Disks Method

The Broadcast Disks method (also known simply as rotating disks) is a generally approved algorithm for data broadcast scheduling in push-based wireless network configurations [4]. This method typically considers one central server transmitting a series of pages in a circular fashion, according to a predefined broadcast program. This program, which will be referred to as the *broadcast sequence*, must be constructed in compliance with two simple rules: first, the number of repetitions of each page in a single broadcast sequence must be proportional to its demand, which is supplied by the network's clients through a feedback mechanism. Second, all instances of the same page must be uniformly distributed inside the broadcast sequence in order to avoid the infamous bus stop paradox. To this end, the pages are grouped according to their popularity in a fixed number of groups called disks. This scheme can be abstractedly represented as an array of physical disks, spinning around a common axis. Each of these disks is then set to spin with an angular velocity proportional to the aggregate demand of its contained pages. An imaginary set of stationary heads then retrieves pages from the disks and forwards them to the broadcasting system in the same order they have been read. Thus, the two aforementioned criteria are met.

One such procedure implies the use of four separate algorithms:

- One to provide feedback on the pages' popularity
- Another to group the pages into disks

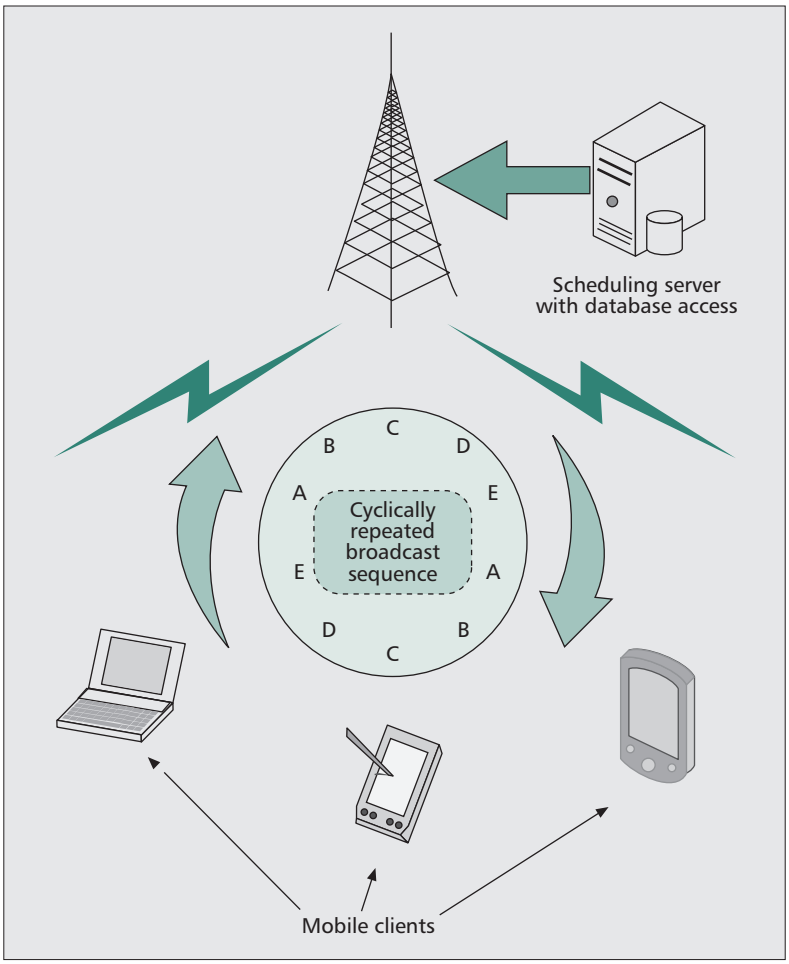


Figure 1. *Network layout.*

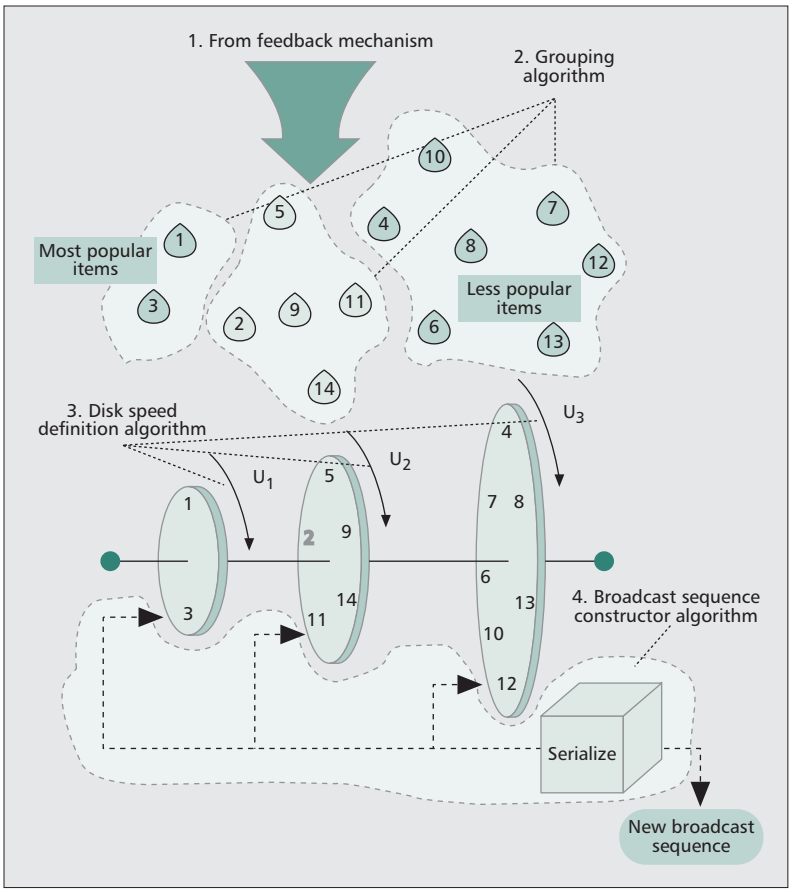


Figure 2. *Broadcast Disks method overview.*

- A third one to define their spinning velocities
- The broadcast sequence constructor algorithm depicted in Fig. 2

Clustering-Driven Wireless Data Broadcasting

While several grouping algorithms have been proposed in the modern literature to date, the number of available spinning velocity definition algorithms is extremely limited; thus, their contribution to the system's overall performance is completely overlooked. In addition, no realistic feedback mechanism has been proposed so far. Moreover, no in-depth (client-wise) performance analysis of complete combinations of such algorithms has ever been made. In realistic scenarios, though, one would anticipate a wide range of different client configurations to be present in the system. Thus, any performance comparison between procedures should take place in this context, and a wide variety of differently parameterized clients should be evaluated. One method, then, should be characterized as better than another only if it is dominant in the majority of these test cases.

The purpose of this article is to introduce the Clustering-Driven Wireless Data Broadcasting (CWDB) procedure, a complete combination of the aforementioned algorithms, comprising a new feedback scheme, a new grouping algorithm, and a new velocity definition algorithm. This combination will be evaluated — as described above — in a wide variety of different client configurations. In all cases, the proposed CWDB will be compared with the GREEDY-Based Procedure (GBP), named after the well-known and efficient grouping algorithm [3] it incorporates. The server's mean response time serves as the system's performance metric and is derived from a simulation of the whole system.

The new feedback algorithm is based on the transmission of a single binary bit of approval when the client is satisfied with the broadcast page. The new grouping method employs the *K*-means clustering algorithm [5], which has been successfully applied in several other network traffic handling algorithms [6, 7]. It also takes into account the fact that the network's clients require only a portion of the total pages to be broadcast by the server at any given time. The same observation is utilized by the new disk velocity definition algorithm. Finally, the broadcast sequence constructor algorithm will be the one first presented in [2].

Network Operation

The network's function is a repeating procedure and is visually represented in Fig. 3. We begin its description from the point where a new broadcast sequence has already been constructed, and an arbitrary page X is being broadcast by the server. If — and only if — this page is actually needed by a client, he/she replies with a single approval message (i.e., one binary digit in a noiseless environment). Thus, the server acquires a posteriori knowledge of the clients' demands, with a minimal client upload rate.

This feedback philosophy is perhaps the only one suited for a push-based system. Any other approach is bound to fall into a client request/server response scheme, which is known to be inappropriate for asymmetric communications, since it would eventually require bigger upload bandwidth from the client and a greater amount of request management effort from the server. The current feedback scheme is based on the one presented in [8].

In practice, a code-division multiple access (CDMA) approach could be adopted for the realization of the client's uplink communication. At first, each client is assigned a unique code, which is commonly referred to as Client Long Code or CLC. Its vote is then encoded by using a high-speed encoding algorithm and sent to the server. At the receiver's end (i.e., the server), signals are separated by means of a correlator (e.g., a rake receiver), which is set to accept signals encoded with the specific CLC only and despread its spectrum. Other clients' signals with different CLCs will remain undecoded and appear as noise. A more detailed description of the CDMA technique and the advantages it offers over other multiple access techniques can be found in [9, 10].

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 broadcast page's registry value. This process is repeated for every broad-

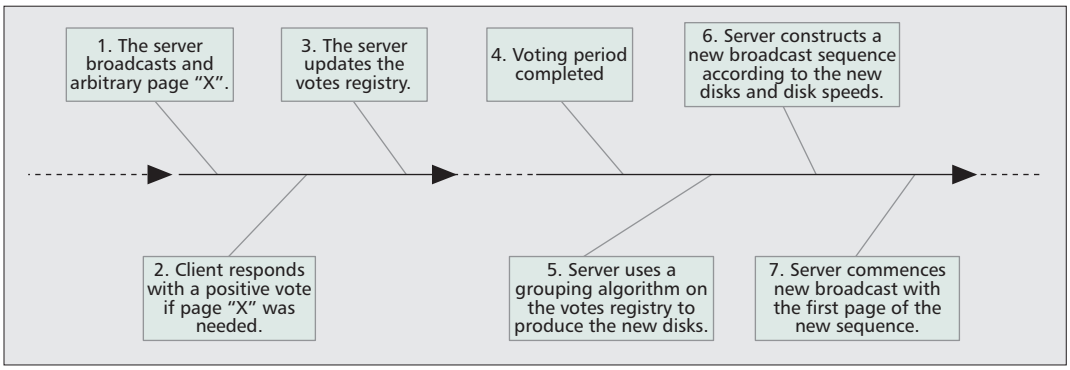


Figure 3. *Feedback mechanism and network function overview.*

cast 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 it has been observed that the real θ parameter of the client's zipf pdf — a parameter controlling the degree of correlation between different data channels — is approximately equal to a constant value, the voting period can be set to last until the calculated θ reaches this value. Actual calculation of this interval is not among the purposes of this article, though. At the end of the voting period, the votes registry has by approximation the form of an unevenly decreasing step function. This is due to the fact that the broadcast pages are assumed to be organized in equiprobable, equally sized groups (i.e., channels), which follow the unevenly decreasing Zipf function.

The next step is to use a grouping algorithm on the completed votes registry in order to create a predefined number of clusters (NoC) containing pages with similar popularity. As mentioned earlier, these clusters represent the well-known Broadcast Disks of the homonymous method. A spinning velocity definition algorithm is then used to assign proper angular velocities to these disks. GBP follows the most common approach, which uses an arbitrary constant value called Δ in order to set the angular velocities of the disks in an arithmetic progression fashion: Δ is the common difference of the sequence and its initial term — which is set to one unit represents the speed of the last and less popular disk. CWDB, on the other hand, considers the speed of the second-to-last disk as another external parameter, and thus uses a modified version of the aforementioned spinning velocity definition algorithm. The reason for this alteration as well as its full description is given further below.

Finally, the broadcast sequence constructor algorithm of [2] is used to create the new broadcast sequence. The votes registry is then nullified, and the procedure repeats itself.

Since CWDB utilizes the *K*-means algorithm to group the pages into disks, it was imperative that we examind this algorithm in greater depth.

Clustering Background

Clustering is an important step in the process of data analysis with applications to numerous fields such as [6, 11]. Informally, clustering is defined as the problem of partitioning data objects into groups (i.e., clusters), such that objects in the same group are "similar," while objects in different groups are "dissimilar." In our framework the data objects to be partitioned are the pages broadcast by the server. For the purpose of clustering, the *K*-means algorithm is employed [5]. *K*-means is a partitional clustering algorithm that produces a certain number of clusters (assume *NoC* clusters) fixed a priori. It is a hill climbing algorithm that is not guaranteed to converge to a global optimum. However, it is widely used because it has low complexity (i.e., linear with the number of pages), is efficient and works very well in practice. The main idea of the *K*-means algorithm is the following: given n points to be clustered, a distance measure d to capture their dissimilarity, and the number of clusters *NoC* to be created, the algorithm initially selects *NoC* random points as clusters' centers and assigns the rest of the $n - NoC$ points to the closest cluster center (according to d). Then, within each of these *NoC* clusters, the cluster representative (also known as centroid or mean) is computed, and the process continues iteratively with these representatives as the new clusters' centers, until convergence. Considering all clusters, the clustering process is guided by the objective function, which is defined to be the sum of distances between each page and the representative of the cluster to which the page is assigned. A particular clustering that optimizes (minimizes) the objective function is the one that groups together pages that present similar popularity among the clients. Once the page clusters are obtained, the CWDB method forms the disks according to clusters' membership as depicted in Fig. 2.

CWDB vs. GBP

As stated earlier, CWDB is a procedure consisting of the feedback mechanism described earlier, a clustering-driven grouping algorithm, a disk velocity definition algorithm, and, finally, a broadcast sequence constructor. In this section we describe the latter three components further.

At the end of the voting period, it is common for a client to have accessed only a portion of the total number of the server's pages. This means that a — usually significant — amount of pages are of no use to the clients. Their exact number is derived from the completed votes reg-

This feedback philosophy is perhaps the only one fitted for a push-based system. Any other approach is bound to fall into a client request/server response scheme, which is known to be inappropriate for asymmetric communications.

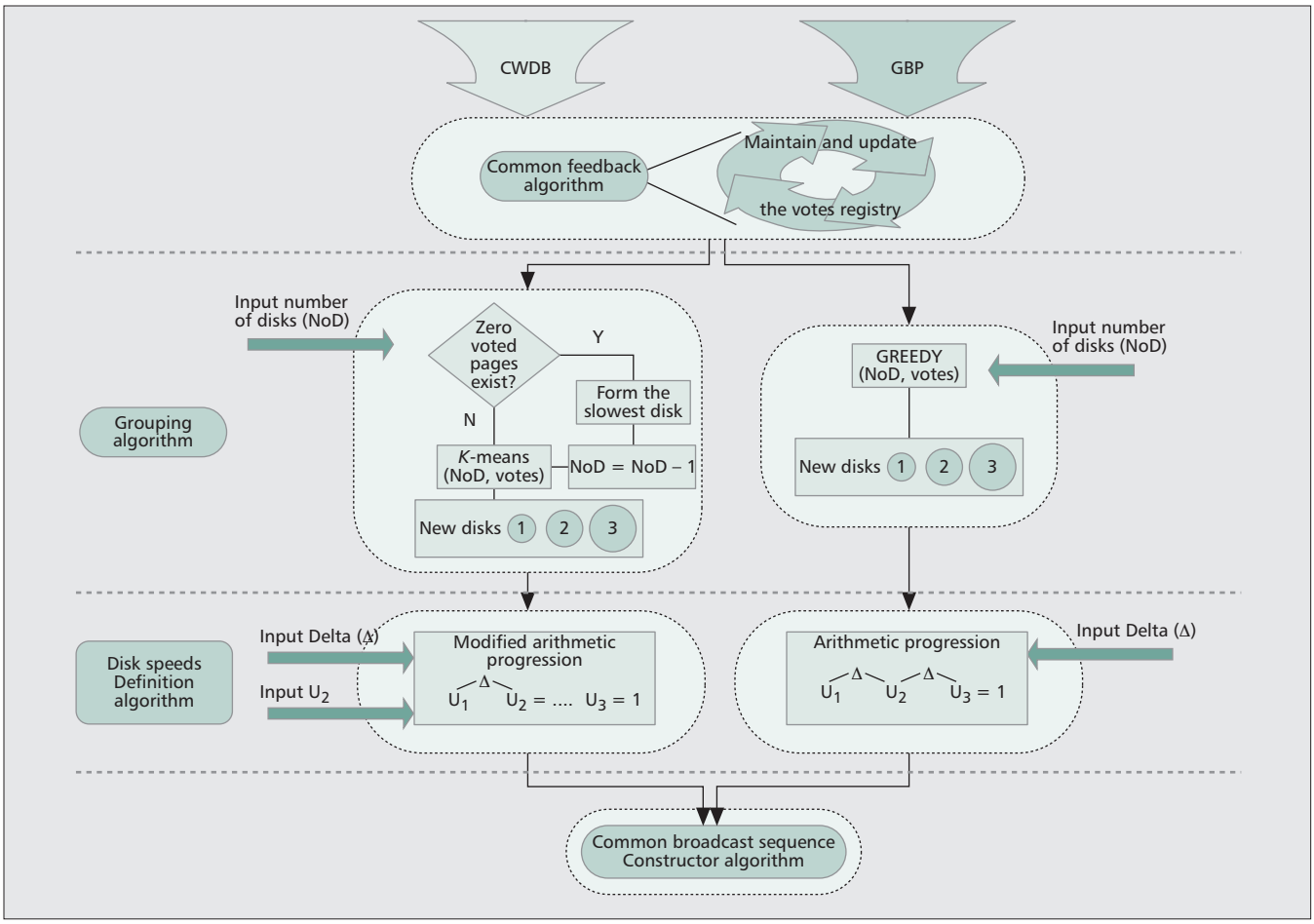


Figure 4. *GBP and CWDB comparison.*

istry by counting the zero voted pages. These items form the last and slowest moving disk whose speed is deterministically set to one unit. The *K*-means algorithm described earlier is applied to the remaining — useful — pages, which are thereby grouped into a predefined number of *NoC* clusters. Thus, the total number of disks *NoD* is equal to *NoC* + 1, and the sizes of the first *NoD* – 1 disks are equal to the number of the items contained in their corresponding clusters.

CWDB states that in order to set the angular velocities of the first *NoD* – 1 disks, the speed of the second-to-last disk $U_{(NoD-1)}$ must be considered as an additional external parameter, just like the Δ parameter of GBP's spinning velocity definition algorithm. The resultant series can be described as a modified arithmetic progression, where the initial term is equal to one unit, the second is defined by the user, and the remaining terms follow the common difference rule. The reason for this alteration is that the $U_{(NoD-1)}$ parameter actually serves as a meter of the difference in importance between the popular and zero voted pages, and is therefore eligible for some fine tuning.

In the case of the GREEDY-based procedure the same feedback scheme as in CWDB is used, but different grouping and spinning velocity definition algorithms are employed: the homonymous GREEDY grouping and spinning velocity definition algorithms described earlier. This is in accordance with [3], where the GREEDY grouping algorithm was first presented. A visual comparison of the two procedures is given in Fig. 4.

Both procedures require a broadcast sequence constructor algorithm as a final step, complying with the two criteria mentioned earlier. In both cases the algorithm first presented in [3] will be used. This algorithm requires the disk sizes and disk velocities (i.e., U_i, which denotes the velocity of the *i*th disk) as inputs, which are supplied by the preceding feedback and spinning velocity definition algorithms.

The goal of the broadcast sequence constructor algorithm is to ensure that the number of repetitions of each page in the final broadcast schedule is equal to the velocity of the disk to which the page belongs. In addition, it ensures that the time interval between two consecutive appearances of the same page is steady throughout the broadcast schedule. This is achieved by splitting each disk into a number of properly sized chunks and broadcasting them in a round-robin manner, as thoroughly described in [2], where this algorithm was first introduced.

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 are then substituted with the most popular of the server's pages.

Simulation Results

CWDB was partially created to address a major drawback of the bibliography on the broadcast disks method so far: the absence of feedback mechanisms and complete combinations of cooperating algorithms. Yet another major omission of the preceding schemes was the lack of thorough testing of the presented algorithms under a wide range of their parameters' values and specifically under various client configurations. As shown in this section, this course of action definitely leads to erroneous estimations of the algorithms' efficiency and the general influence of the various parameters on the system's performance. We therefore tested the two aforementioned procedures under a wide range of parameter values given in Table 1.

Values of θ smaller than one unit correspond to a high degree of correlation between channels, while values in the range (1, ∞) suggest the opposite. The values 0.3 and 1.5 represent the two corresponding extremes. The value θ =1 is not allowed due to a Zipf pdf restriction. The value 0.95 is the most commonly used in similar work.

The *ClientRange* parameter represents the number of pages the client has accessed at the end of a voting period. The total number of available pages was set to 5000. The *ChannelSize* parameter obviously denotes the number of pages in a single channel. These ranges yield a total number of 12,960 runs on the simulator.

For each case we started with a null votes registry. The voting period was set to last for 35,000 client queries (in one client model). For an additional amount of 15,000 client queries we observed the server's response time. We assumed that the client's processing time is equal to two time slots. In other words, upon receiving a requested page the client remains inactive for a period of two broadcast time slots. Finally, we recorded the mean response time and the produced disk sizes.

The results are displayed in Table 2. Each row represents a specific *ClientRange*/*ChannelSize* pair of values, while the column names denote the corresponding values of θ of the client's Zipf pdf. Thus, each table cell represents a unique client configuration case. For each one of these cases, we execute 100 different simulation runs for every possible combination of the Δ, $U_{(NoD-1)}$ and *NoD* parameters, presented in Table 2, and calculate the mean response times. Due to the vast number of results and in order to better compare CWDB and GBP, we present only the smallest achieved mean response time for each algorithm, accompanied by the corresponding Δ, $U_{(NoD-1)}$ in case of CWDB and only Δ in case of GBP, and the resulting disk configuration for each of these cases. Moreover, a 95 percent confidence interval is supplied for the presented mean response times. Space restrictions required a certain way of formatting the aforementioned presented values. The line in bold denotes the dominant algorithm in each case (i.e., the one achieving the smallest mean response time).

The reader may notice slight variations in the response time in some cases that share the same Δ, $U_{(NoD-1)}$ and disk configuration (e.g., cells (1, 1) and (1, 2) of Table 2). 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); thus, variations of ±15 time units have been observed and should be considered natural.

θ	0.3, 0.5, 0.7, 0.95, 1.1, 1.5
Δ	1, 2, 3, 4, 5, 6, 7, 8, 15, 20, 50, 80, 100
PageRange/*ChannelSize*	30/1, 30/3, 30/5, 1000/30, 1000/50 2000/50, 2000/100, 3000/50, 4000/50
$U(N_{oD-1})$ (CWDB only)	10, 30, 50, 100
N_{oD}	2, 3, 4, 5, 6, 7

Table 1. *Simulation parameters.*

In some cases the zero voted pages are more than expected. For example, for *ClientRange* = 3000 we would expect the slowest disk of all CWDB configurations to contain exactly 2000 pages. Yet one can easily see that values like 2012, 2045, and even 2616 have appeared. This is also a consequence of the aforementioned randomness of the simulation model. In accordance with the Zipf pdf, certain pages have an extremely low probability of being requested by the client. This phenomenon becomes progressively more intense as θ increases. For example, for θ = 1.5 the access probability of the 5000th page is $\sim 10^{-6}$. Thus, even after an enormous voting period of 35,000 client queries, a certain number of pages will probably have not even been requested (or voted for) once. As a consequence the actual *ClientRange* may usually be slightly bigger than expected.

The aforementioned phenomenon prohibited the use of *ClientRanges* > 4000 pages during the simulation. This must not be considered an omission though. CWDB requires the existence of an adequate number of unused pages and should therefore not be used under these conditions anyway. Additionally, it is highly improbable that the clients will actually need all of the server's pages at any given time.

The accuracy of the above simulation results can be verified by their confidence intervals. More specifically, the 95 percent confidence intervals of the smallest achieved mean response time in case of CWDB and GBP are computed and presented in Table 2.

Finally, in some cases GBP gave its response time for Δ = 15, which was the upper bound of the value set for this parameter during the simulation. Assuming that higher values of Δ could improve GBP's performance, we tested higher values as well, e.g., Δ = 20, 50, 80, 100, especially for these cases.

Major Observations

The following conclusions can be extracted from the simulation results presented in Table 2:

- CWDB outperforms GBP in the vast majority of the test cases with the difference in performance increasing as *ClientRange* decreases. This

ClientRange / ChannelSize			θ = 0.3	θ = 0.5	θ = 0.7	θ = 0.95	θ = 1.1	θ = 1.5
30 / 1	CWDB	Response Time	**39 ± 1.97%**	**37 ± 2.19%**	**36 ± 2.28%**	**35 ± 2.74%**	**37 ± 2.30%**	**32 ± 5.00%**
		Δ,UN-1	**1,100**	**1,100**	**3,100**	**4,100**	**1,100**	**2,100**
		Disk Sizes	**[4 26 4970]**	**[4 26 4970]**	**[3 27 4970]**	**[3 27 4970]**	**[4 26 4970]**	**[1 2 27 4970]**
	GBP	Response Time	98 ± 1.61%	95 ± 2.32%	93 ± 2.86%	98 ± 3.04%	91 ± 3.53%	100 ± 4.56%
		Δ	50	50	50	50	50	50
		Disk Sizes	[13 17 4970]	[12 18 4970]	[10 20 4970]	[9 21 4970]	[12 18 4970]	[8 22 4970]
30 / 3	CWDB	Response Time	**33 ± 0.01%**	**34 ± 1.65%**	**35 ± 1.43%**	**35 ± 1.34%**	**37 ± 0.73%**	**27 ± 4.00%**
		Δ,UN-1	**2,100**	**1,100**	**4,100**	**2,100**	**1,100**	**4,100**
		Disk Sizes	**[12 18 4970]**	**[2 10 18 4970]**	**[5 25 4970]**	**[4 26 4970]**	**[4 26 4970]**	**[4 26 4970]**
	GBP	Response Time	101 ± 1.61%	88 ± 2.05%	98 ± 1.81%	98 ± 2.07%	81 ± 2.98%	89 ± 2.98%
		Δ	50	50	50	50	50	50
		Disk Sizes	[12 18 4970]	[12 18 4970]	[11 19 4970]	[10 20 4970]	[12 18 4970]	[7 23 4970]
30 / 5	CWDB	Response Time	**29 ± 2.59%**	**31 ± 2.94%**	**35 ± 0.01%**	**33 ± 1.94%**	**34 ± 2.29%**	**30 ± 2.80%**
		Δ,UN-1	**3,100**	**2,100**	**2,100**	**3,100**	**1,100**	**2,100**
		Disk Sizes	**[18 12 4970]**	**[3 15 12 4970]**	**[6 24 4970]**	**[8 22 4970]**	**[3 5 22 4970]**	**[3 5 22 4970]**
	GBP	Response Time	103 ± 1.56%	100 ± 1.55%	89 ± 1.65%	91 ± 2.29%	89 ± 1.88%	95 ± 2.45%
		Δ	50	50	50	50	50	50
		Disk Sizes	[13 17 4970]	[13 17 4970]	[12 18 4970]	[11 19 4970]	[10 20 4970]	[8 22 4970]
1000 / 30	CWDB	Response Time	**517 ± 0.72%**	**502 ± 0.39%**	**471 ± 0.34%**	**430 ± 0.30%**	**409 ± 0.33%**	**364 ± 0.94%**
		Δ,UN-1	**5,100**	**1,100**	**1,100**	**1,100**	**1,100**	**3,30**
		Disk Sizes	**[1000 4000]**	**[131 869 4000]**	**[30 157 813 4000]**	**[28 93 875 4004]**	**[28 96 875 4001]**	**[28 60 902 4010]**
	GBP	Response Time	666 ± 0.70%	695 ± 1.32%	766 ± 1.24%	839 ± 1.08%	866 ± 1.44%	720 ± 0.70%
		Δ	15	15	6	5	3	7
		Disk Sizes	[990 4010]	[386 592 4022]	[331 618 4051]	[87 165 606 4142]	[67 130 546 4257]	[106 390 4504]
1000 / 50	CWDB	Response Time	**518 ± 0.60%**	**506 ± 0.41%**	**480 ± 0.31%**	**439 ± 0.34%**	**415 ± 0.34%**	**356 ± 0.31%**
		Δ,UN-1	**4,100**	**1,100**	**1,100**	**1,100**	**1,100**	**2,100**
		Disk Sizes	**[1000 4000]**	**[182 818 4000]**	**[50 184 766 4000]**	**[48 155 797 4000]**	**[48 148 804 4000]**	**[48 93 857 4002]**
	GBP	Response Time	696 ± 0.51%	659 ± 1.07%	712 ± 1.00%	845 ± 1.28%	862 ± 0.68%	779 ± 0.70%
		Δ	50	15	15	6	5	6
		Disk Sizes	[996 4004]	[398 589 4013]	[347 620 4033]	[275 602 4123]	[228 582 4190]	[141 426 4433]
2000 / 50	CWDB	Response Time	**1001 ± 0.61%**	**991 ± 0.38%**	**944 ± 0.32%**	**849 ± 0.29%**	**836 ± 0.22%**	**885 ± 0.23%**
		Δ,UN-1	**1,100**	**1,100**	**1,100**	**1,100**	**2,30**	**1,30**
		Disk Sizes	**[2000 3000]**	**[248 1751 3001]**	**[75 274 1649 3002]**	**[48 170 1777 3005]**	**[48 147 1794 3011]**	**[48 101 1675 3176]**
	GBP	Response Time	1867 ± 0.57%	1871 ± 0.63%	1710 ± 1.13%	1540 ± 1.20%	1361 ± 1.27%	969 ± 1.42%
		Δ	1	1	1	1	2	3
		Disk Sizes	[1390 3610]	[1390 3610]	[34 41 103 280 810 3732]	[23 25 68 208 750 3869]	[92 166 654 4088]	[50 98 457 4395]
2000 / 100	CWDB	Response Time	**1025 ± 0.58%**	**999 ± 0.42%**	**955 ± 0.37%**	**870 ± 0.33%**	**850 ± 0.32%**	**902 ± 0.27%**
		Δ,UN-1	**1,100**	**1,100**	**1,100**	**1,100**	**1,100**	**2,30**
		Disk Sizes	**[2000 3000]**	**[375 1621 3004]**	**[250 1749 3001]**	**[98 300 1601 3001]**	**[99 289 1608 3004]**	**[98 191 1625 3086]**
	GBP	Response Time	1870 ± 0.58%	1870 ± 0.57%	1778 ± 0.41%	1607 ± 0.57%	1501 ± 1.25%	1146 ± 1.20%
		Δ	1	1	1	1	2	3
		Disk Sizes	[1562 3474]	[1452 3548]	[44 74 114 294 816]	[66 82 233 750 3869]	[119 195 693 3993]	[84 124 543 4249]
3000 / 50	CWDB	Response Time	**1503 ± 0.62%**	**1475 ± 0.39%**	**1441 ± 0.41%**	**1355 ± 0.29%**	**1273 ± 0.22%**	1056 ± 1.77%
		Δ,UN-1	**1,100**	**1,30**	**1,10**	**2,10**	**2,10**	3,10
		Disk Sizes	**[3000 2000]**	**[334 2666 2000]**	**[92 526 2370 2012]**	**[48 216 2691 2045]**	**[48 198 2625 2129]**	[48 105 2231 2616]
	GBP	Response Time	2200 ± 0.70%	2152 ± 0.82%	2006 ± 0.90%	1696 ± 1.23%	1551 ± 1.26%	**1037 ± 1.21%**
		Δ	1	1	1	1	1	**3**
		Disk Sizes	[1703 3297]	[1507 3493]	[191 306 875 3628]	[24 25 72 215 770 3894]	[94 170 673 4063]	**[50 98 462 4390]**
4000 / 50	CWDB	Response Time	**1999 ± 0.65%**	**1997 ± 0.73%**	**1896 ± 0.37%**	**1795 ± 0.25%**	1729 ± 1.86%	1297 ± 0.13%
		Δ,UN-1	**1,100**	**1,100**	**1,10**	**2,10**	2,10	3,10
		Disk Sizes	**[3991 1009]**	**[3990 1010]**	**[80 488 3484 948]**	**[49 209 3603 1139]**	[48 200 3428 1324]	[48 115 2600 2237]
	GBP	Response Time	2378 ± 0.46%	2274 ± 0.45%	2141 ± 0.55%	1797 ± 0.86%	**1605 ± 1.22%**	**1061 ± 1.18%**
		Δ	1	1	1	1	**1**	**3**
		Disk Sizes	[1858 3142]	[1726 3274]	[1482 3918]	[52 77 239 843 3798]	**[20 21 57 182 746 3974]**	**[49 97 469 4385]**

Table 2. *Simulation results.*

was expected as CWDB relies on the existence of a relatively large amount of unused pages.

- In the cases where both CWDB and GBP produce similar disk configurations, the difference in performance can only be ascribed to the new velocity definition algorithm introduced in this article. This outcome confirms that the second to last disk is an important factor in the system's performance. This fact also denotes the need for further research on other speed definition schemes apart from arithmetic-progression-based ones.
- The optimal number of disks is usually two, three, or four. Three-disk configurations are the most common. Values higher than four are extremely rare.
- The *K*-means clustering algorithm employed by CWDB tends to perform a more selective grouping of pages than GBP. The faster disks produced by CWDB usually contain fewer pages than their GBP counterparts. We are currently researching the impact of both tactics on the system's behavior from a bottleneck avoidance aspect.
- In all but a few cases CWDB produces the response time for $\Delta = 1$ and $U_{(NoD-1)} = 100$. Thus, there is no real need to create an optimizing scheme regarding these parameters. On the other hand, the optimal Δ for GBP ranges from 1 to 50, making the need for such a scheme imperative.
- It is clear that the *ClientRange* parameter greatly affects the response time. Thus, it is obvious that every broadcast scheduling procedure must be tested against a variety of client configurations in order to evaluate its performance in a realistic manner.
- The θ parameter affects the response time indirectly by defining the optimal duration of the

voting period. This is why its effect is trivial for small client ranges but increases for bigger values of this parameter. The actual computation of the optimal duration of the voting period is also a work in progress. Future steps also include the development of a more realistic multiclient simulation model.

Conclusions

In the context of the present work, a new clustering-driven wireless data broadcasting procedure has been introduced and tested thoroughly. The detailed analysis that took place clarified the performance impact of nearly every parameter of the system. Performance comparison with other classical solutions suggests that the use of clustering algorithms can be the basis of a new generation of high-performance data broadcasting schemes.

References

[1] P. Nicopolitidis *et al.*, "Performance Optimization of an Adaptive Wireless Push System in Environments with Locality of Demand," *Comp. Commun.*, vol. 29, no. 13–14, 2006, pp. 2542–49.

[2] S. Acharya *et al.*, "Broadcast Disks: Data Management for Asymmetric Communication Environments," *Proc. ACM SIGMOD '95*, New York, May 1995.

[3] W. Yee *et al.*, "Bridging the Gap between Response Time and Energy-Efficiency in Broadcast Schedule Design," *Proc. 8th Int'l. Conf. Extending Database Tech.*, London, U.K., 2002.

[4] S. Acharya, M. Franklin, and S. Zdonik, "Dissemination-based Data Delivery Using Broadcast Disks," *IEEE Wireless Commun.*, vol. 2, no. 6, 1995, pp. 50–60.

[5] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer, 2001.

[6] S. Petridou *et al.*, "On the use of Clustering Algorithms for Message Scheduling in WDM Star Networks," *IEEE J. Lightwave Tech.*, to appear, 2008.

[7] S. Petridou *et al.*, "A Clustering-Driven Medium Access Control Protocol for WDM Star Networks," *Optics & Laser Technology*, vol. 41, no. 1, 2009, pp. 42–52.

[8] P. Nicopolitidis, G. Papadimitriou, and A. Pomportsis, "Using Learning Automata for Adaptive Push-Based Data Broadcasting in Asymmetric Wireless Environments," *IEEE Trans. Vehic. Tech.*, vol. 51, no. 6, 2002, pp. 1652–60.

[9] W. Lee, "Overview of Cellular CDMA," *IEEE Trans. Vehic. Tech.*, vol. 40, no. 2, 1991, pp. 291–302.

[10] K. Gilhousen *et al.*, "On the Capacity of A Cellular CDMA System," *IEEE Trans. Vehic. Tech.*, vol. 40, no. 2, 1991, pp. 303–12.

[11] S. Petridou *et al.*, "Time Aware Web Users Clustering," *IEEE Trans. Knowledge and Data Eng.*, vol. 20, no. 5, 2008, pp. 653–67.

Biographies

CHRISTOS K. LIASKOS received his Diploma degree in electrical and computer engineering in 2004 and his M.S. degree in medical informatics in 2008 from the Aristotle University of Thessaloniki, Greece. He is currently working toward his Ph.D. degree in communication networks. His research interests include wireless networks, neural networks and fractal compression algorithms.

SOPHIA G. PETRIDOU [M'08] received her Diploma and Ph.D. degrees in computer science from the Aristotle University of Thessaloniki, Greece, in 2000 and 2008, respectively. Her research interests include clustering, optical, and wireless networks.

GEORGIOS I. PAPADIMITRIOU [M'89, SM'02] received his Diploma and Ph.D. degrees in computer engineering from the University of Patras, Greece, in 1989 and 1994, respectively. In 1997 he joined the faculty of the Department of Informatics, Aristotle University of Thessaloniki, where he is currently serving as an associate professor. His main research interests include optical networks and wireless networks. He is an Associate Editor of five IEEE journals. He is co-author of three books published by Wiley. He is author or coauthor of 80 journal and 90 conference papers.

PETROS NICOPOLITIDIS received B.S. and Ph.D. degrees in computer science from the Department of Informatics, Aristotle University of Thessaloniki, in 1998 and 2002, respectively. Since 2004 he is a lecturer with the same department. His research interests are in the areas of wireless networks and mobile communications. He is coauthor of the book *Wireless Networks* (Wiley, 2003).

MOHAMMAD S. OBAIDAT [S'85, M'86, SM'91, F'05] (obaidat@monmouth.edu) is an internationally known academic, researcher, and scientist. He received his Ph.D. and M.S. degrees in computer engineering with a minor in computer science from The Ohio State University, Columbus. He is currently a full professor of computer science at Monmouth University, New Jersey. Among his previous positions are chair of the Department of Computer Science and director of the Graduate Program at Monmouth University, and faculty member at the City University of New York. He has received extensive research funding. He has authored or co-authored 10 books and over 425 refereed scholarly journal and conference articles. He has served as a consultant for several corporations and organizations worldwide, and is editor of many scholarly journals, including Editor-in-Chief of the *International Journal of Communication Systems* published by Wiley. He is also an Editor of *IEEE Wireless Communications*. In 2002 he was the scientific advisor for the World Bank/UN Workshop on Fostering Digital Inclusion. Recently, he was awarded the distinguished Nokia Research Fellowship and the Distinguished Fulbright Award. He has made pioneering and lasting contributions to the multifaceted fields of computer science and engineering. He has guest edited numerous special issues of scholarly journals such as *IEEE Transactions on Systems, Man and Cybernetics*, *IEEE Wireless Communications*, *IEEE Systems Journal*, and *Elsevier Performance Evaluation*, among others. He has served as Steering Committee Chair, Advisory Committee Chair, Honorary Chair, and Program Chair of many international conferences. He is the founder of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS) and has served as General Chair of SPECTS since its inception. He has received a recognition certificate from the IEEE. Between 1994–1997 he served as a distinguished speaker/visitor of IEEE Computer Society. Since 1995 he has been serving as an ACM Distinguished Lecturer. He is also an SCS Distinguished Lecturer. He is the founder of the SCS Distinguished Lecturer Program (DLP) and its present director. Between 1996 and 1999 he served as an IEEE/ACM program evaluator of the Computing Sciences Accreditation Board/Commission, CSAB/CSAC. Between 1995 and 2002 he served as a member of the Board of Directors of the Society for Computer Simulation International. Between 2002 and 2004 he served as Vice President of Conferences of the Society for Modeling and Simulation, International SCS. Between 2004 and 2006 he served as Vice President of Membership of SCS, and from 2006 to 2009 he served as Senior Vice President of SCS. Currently, he is President of SCS. He has been invited to lecture and give keynote speeches worldwide. His research interests are wireless communications and networks, modeling and simulation, performance evaluation of computer systems and telecommunications systems, security of computer and network systems, high-performance computers, applied neural networks and pattern recognition, security of e-based systems, and speech processing. During the 2004–2005 academic year he was on sabbatical leave as the Fulbright Distinguished Professor and Advisor to the President of Philadelphia University (Dr. Adnan Badran, who in April 2005 became the Prime Minster of Jordan). He is a Fellow of the Society for Modeling and Simulation International SCS.

ANDREAS S. POMPORTSIS received his B.S. degree in physics and M.S. degree in electronics and communications, both from the University of Thessaloniki, Greece, and his Diploma in electrical engineering from the Technical University of Thessaloniki, Greece. In 1987 he received his Ph.D. degree in computer science from the University of Thessaloniki. Currently, he is a professor in the Department of Informatics, Aristotle University, Thessaloniki, Greece. He is co-author of the books *Optical Switching* (Wiley, 2007), *Multiwavelength Optical LANs* (Wiley, 2003), and *Optical Networks* (Wiley, 2003). His research interests include computer networks, computer architecture, parallel and distributed computer systems, and multimedia systems.