# Broadcasting Images in Wireless Networks

Alexandros Nanopoulos
Aristotle University
ananopou@csd.auth.gr

Athanasios Nikolaidis
Tech. & Edu. Inst. of Serres
nikola@delab.csd.auth.gr

Apostolos Papadopoulos
Aristotle University
apostol@delab.csd.auth.gr

## Abstract

*Server-initiated broadcast, compared to unicast transmission, presents excellent scalability to requests by multiple clients in wireless networks. In such networks, many mobile clients can have overlapping interests about the same visual information, thus image broadcasting is expected to find acceptance in wireless broadcast networks (WBNs). In this paper we examine for the first time, to our knowledge, the problem of image broadcasting in WBNs. We propose a novel method that significantly reduces the time latency perceived by clients that request images. We also consider the issue of energy consumption, because portable devices operate with batteries. With experimental results we verify the superiority of the proposed method against existing methods from other domains.*

## 1 Introduction

Advances in wireless technology have brought the demand for visual communication, consisting of images (or video), over wireless networks. Nowadays several formats of wireless images exist, e.g., WBMP (wireless bitmap) and OTB (on-the-air bitmap). Moreover, novel transmission schemes have been proposed [4, 8, 11], which focus on addressing the challenges posed by the wireless medium to the problem of error correction.

There are two basic delivery methods in wireless networks: (a) the unicast (a.k.a. point-to-point or pull-based) transmission and (b) the server-initiated broadcast (a.k.a. push-based). With the former, each mobile client establishes a separate connection with the server (through an uplink channel) and poses a request. Thus, excessive network traffic may incur for "hot" pieces of information, since multiple requests are produced for them. Unicast is the only delivery method that has been examined so far for image transmission in wireless networks [4, 8, 11]. Since image data are relatively large compared to, e.g., text, unicast image transmission can significantly aggravate network loads and reduce the quality of service.

In contrast, with the latter method, all clients monitor the same channel (downlink channel) to acquire data transmitted by the server. Therefore, broadcast is advantageous because of its excellent scalability, as it permits simultaneous requests by numerous clients to be served efficiently. Moreover, it exploits the communication asymmetry in wireless environments; that is, the uplink channel capacity is much smaller than that of the downlink. Several commercial broadcast services have been deployed so far, e.g., StarBand, Hughes Network Systems, and DirectBand Network.[1] They only support plain data types (text or numerical), which have been successful for applications about weather or financial information. However, improvements in wireless technology resulted to bandwidths that can, now, sustain visual communication with image data.[2] Nevertheless, until now, these increasing capabilities have not been explored in depth for image transmission.

### 1.1 Motivation

The first question that follows from the previous discussion is whether images are useful in wireless broadcast networks (WBNs). To answer this question we have to consider that, in several cases, mobile clients have overlapping interests about the same visual information. For example, an image can represent the map of an area and several clients may be interested in receiving it at their PDAs (location-based service). The map may be additionally overlayed with information like the traffic load (e.g.,

[1] www.starband.com, www.microsoft.com/resources/spot, and www.direcway.com, respectively.

[2] For instance, GPRS currently provides bandwidth higher than 64 KBps.

streets annotated with an indicative color). This creates the additional need to update such time-varying images. In another example, images can be associated with headline news. Since these news interest many users, the corresponding images can be broadcast to them along with the textual description. In all the aforementioned examples, it is more efficient to broadcast the requested images than having each client burden the wireless network with separate requests. Therefore, due to their usefulness, it is likely that images are going to become a common type for data dissemination in WBNs, like text currently is.

The second question that follows is why existing (unicast) methods for transmitting images do not suffice in the case of WBNs. Compared to unicast transmission, image broadcast has to consider two new issues: (i) Transmission does not start when the mobile client issues a request, but when the available information is scheduled to be broadcast. (ii) The broadcast channel presents limited capacity (compared to, e.g., a wireless LAN) and higher economic charge. Both issues render unicast methods inapplicable. In particular, the former (i) issue introduces the need to minimize user-perceived latency (waiting time). The latter (ii) issue compels users to request for approximations instead of the original image, whereas each user can set the desired level of approximation depending on the provided bandwidth and/or economic charge.[3] The aforementioned issues have not been examined so far. What is, therefore, required is a novel approach, which will address both these issues.

### 1.2 Contribution

In this paper we examine for the first time, to our knowledge, the problem of image broadcasting in WBNs. We propose a novel scheduling method that generates non-flat programs for image broadcast. Its objective is to minimize the time latency that is perceived by clients when they request an image from a WBN. Additionally, we consider the resulting energy consumption, since portable devices operate with batteries.

The proposed method has the following characteristics:

- It can work with encoding techniques that support progressive image refinement, thus can easily be applied to several widely accepted standards like the progressive mode or hierarchical mode employing progressive coding mode of JPEG [10] or the JPEG2000.
- It exploits features that these standards present, like energy compaction, to identify the elements that should be given higher priority. The identification is done analytically, through the optimization of estimation functions that are developed.
- The generated programs (non-flat) assign higher broadcast frequency to higher priority elements. This leads to significant performance improvement compared to existing methods, which is verified by experimental results.

The rest of this paper is organized as follows. In Section 2 we describe the background and related work. The proposed problem and method are detailed in Sections 3 and 4. The experimental results are given in Section 5, whereas Section 6 concludes the paper.

## 2 Background and related work

### 2.1 Background

In the rest of the paper we assume the following. There exists a fixed *mobile support station* (server), which broadcasts over a single broadcast layer at a constant rate. The data are reliably received by the clients that are within the server's radio coverage. Note that the problem of error correction is orthogonal to our research (see Section 2.2). The physical unit of broadcast data is denoted as *bucket*. Thus, when a client reads data from the channel, the minimum number of bytes it can read each time, is equal to the bucket size.[4] In real applications, the bucket usually has a relatively low size, e.g., 512 bytes or 1 K.

The most critical factor is the scheduling algorithm, which organizes the periodically broadcast items in a *program* that determines their order and frequency (i.e., number of appearances) in each cycle. Each program contains information about an image that is encoded with an existing standard like JPEG or JPEG2000. For simplicity, we will assume JPEG, but our approach can be easily generalized to JPEG2000.

In JPEG, the image is partitioned into disjoint windows of size $8 \times 8$ pixels. Discrete Cosine Transform (DCT) is applied within each window, producing a sequence of 64 DCT coefficients that are ordered. In particular, the DC coefficient is of order 0, whereas the order of the highest

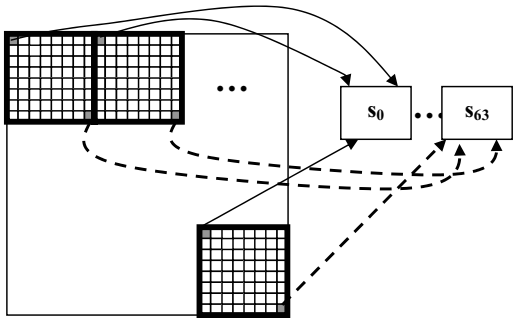
[3] Approximations are sufficient for the additional reason that light mobile devices (e.g., PDAs) usually have screens with relatively low resolution.

[4] The notion of bucket is analogous to a physical page in a hard disk device.

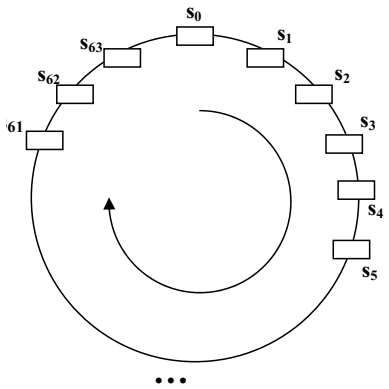
frequency AC coefficient is 63. Finally, the coefficients of each window are quantized and encoded (e.g., run-length or huffman). We collect the quantized and encoded DCT coefficients from all $8 \times 8$ windows that are of the same order, $i$, into a new sequence $C_i$, $0 \leq i \leq 63$. The contents (coefficients) of each sequence $C_i$ are then placed into a series of buckets (because they may not fit in only one bucket). Such a sequence of buckets that contains the contents of $C_i$ is denoted as *segment* $s_i$, where $|s_i|$ denotes the number of buckets required by $s_i$. Figure 1a illustrates the formation of the resulting 64 segments. In contrast to a bucket, which is the physical unit of transmission, a segment acts as the logical unit of transmission, since a client has to read the contents of an entire segment in order to retrieve the value of a specific coefficient within it. In general, as coefficients are encoded, it may hold that $|s_i| \neq |s_j|$ for $i \neq j$. Therefore, as *segment size* we consider the maximum $|s_i|$ for all $0 \leq i \leq 63$, and we assume that, when necessary, all other segments are padded to fit this maximum size.

Using a *flat* program, the segments are just sorted in increasing order of their coefficients and are periodically broadcast one after the other. An example is depicted in Figure 1b. This implements the case of unicast.

Mobile clients can set the desired level of image's approximation (i.e., quality). The approximation level determines the maximum order $m$ of coefficients that will be retrieved, $0 \leq m \leq 63$. After a request, the client waits until $s_0$ starts being broadcast and retrieves all segments $s_i$ for $0 \leq i \leq m$. We define as *access latency*, $A$, the time that the client waits until segment $s_m$ is received. For convenience, we measure $A$ in terms of the corresponding number of segments. For instance, if a request initiates when segment $s_{62}$ is about to be broadcast and $m = 5$, then $A = 8$. Finally, we assume the existence of mechanisms [1, 5] that allow the server to learn statistics, like the request probabilities of the segments.




(a)




(b)

**Figure 1. Example of: (a) mapping from $8 \times 8$ windows to 64 segments; (b) a flat program.**

## 2.2 Related work

Using a flat program, an image is transmitted by simply conveying its segments one after the other, regardless of their access probability. However, some segments are more frequently requested than others, depending on the approximation level that is set by each client. To minimize latency, we have to take into account the access probability $P(s_i)$ for each segment $s_i$. Therefore, segments with higher $P(s_i)$ value should be broadcast more frequently so as to reduce the time clients wait for them. In this case, the resulting program is denoted as non-flat.

Non-flat programs for plain data (e.g., numerical) have been introduced by Acharya et al. [1], whereas subsequent works [2, 6] described ways to determine the frequency of the items in the program. These works examine the retrieval of a single item and are not efficient for image data, because the latter requires multi-item retrieval. For purposes of comparison, however, we examine the adaptation of the MAD scheduling algorithm [6], which is one of the best scheduling algorithms for single-item retrieval.

Scheduling for multi-item retrieval has been considered in [3, 9]. However, the access pattern in these works is different from the one considered in our work. The reason is that [3, 9] concern broadcast items that are "linked" together (e.g., inline images within a web-page), thus consider programs where some items are accessed together

with probability equal to one. In contrast, for image broadcasting, items are accessed together with probability determined by the required quality, which is user-defined. Moreover, in [3, 9] there exist no order in the broadcast items, whereas in image broadcasting, items are ordered with respect to a transformation (e.g., discrete cosine). Program-generation algorithms for retrieving numerical values in specified ranges, are examined by Tan et al. [7]. In the case of image data, ranges start from the first segment and are determined by the required quality. This results to a very different access pattern compared to [7].

Finally, related work includes channel-coding methods for error resilience [4, 8, 11]. They focus on unicast image transmission in wireless networks. Channel-coding methods are orthogonal to our research. Referring to the 7-layer OSI model, we focus on the *Transport* layer, assuming transparency to error-correction mechanisms provided by the *Network* and *Data Link* layers.

## 3 Problem definition

The quality of the received image depends on the maximum order, $m$, of coefficients that will be retrieved. For the example of Lena image, Figure 2 illustrates Mean Square Error (MSE) versus $m$. In general, as also verified by Figure 2, due to the energy-compaction property, MSE reduces rapidly after the first few values of $m$. However, after a point, the reduction becomes much less steep. It is expected that most clients will not be willing to set $m$ higher than this point, as the corresponding image quality is adequate and the additional cost (time and/or economical) does not payoff.

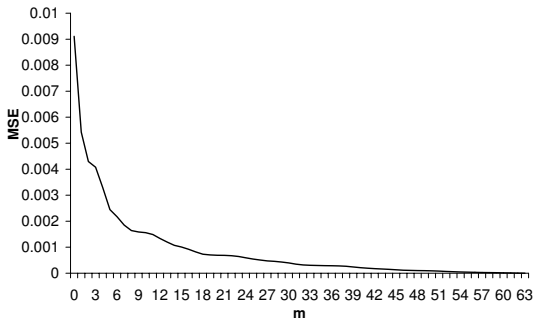



**Figure 2. Example of MSE versus $m$.**

Since different clients will use different values for $m$, each segment will be retrieved with different probability. An observation we have to make is that, if we set $m = i$, we have to retrieve all segments $s_j$ for $0 \leq j \leq i$. For this reason we define the following:

**Definition 1** *For each segment $s_i$ ($0 \leq i \leq 63$), we define the access probability $P(s_i)$ as the normalized number of times that clients requested retrieval up to $s_i$.*

In other words, $P(s_i)$ denotes the probability that a client will set $m = i$. For the reasons mentioned in the beginning of this section (reduction of MSE with increasing $m$), the distribution of $P(s_i)$ is expected to have shape similar to that in Figure 3, which can be approximatively divided into three main parts:

- *Part 1* corresponds to the clients that stop at the very first segments (i.e., use a low $m$ value). Their number is small, as these segments produce low image quality.
- *Part 2* contains the majority of clients, which are those that use a satisfactory $m$.
- *Part 3* corresponds to the clients that use a high $m$ value. In this part $P(s_i)$ reduces, because few clients request such higher-level approximations. The reduction continues until it reaches a minimum, whereas beyond this point $P(s_i)$ remains about constant.

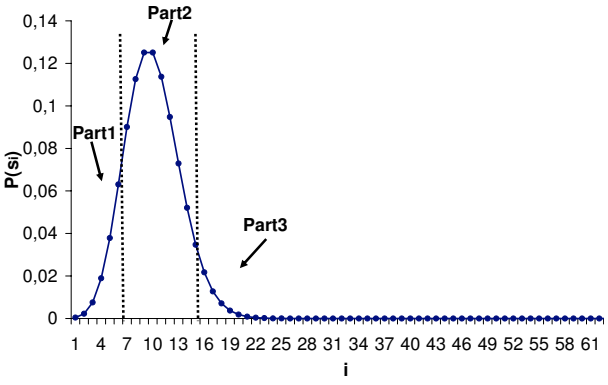



**Figure 3. Example of distribution of $P(s_i)$.**

We have examined various synthetic probability distribution functions for $P(s_i)$, like the Poisson distribution or the Log-normal distribution, which are used to model skewed phenomena. Our experiments indicated that the relative performance of the methods we examined is not changed by the choice of the probability distribution function. For reasons of simplicity, and also because $P(s_i)$ can be considered as a discrete random variable, we henceforth assume that $P(s_i)$ follows the Poisson distribution, that is, $P(s_i) = e^{-\lambda}\lambda^i/i!$.

The mean and variance are equal to $\lambda$. Thus, $\lambda$ determines the shape of the distribution.

To construct efficient non-flat program, we have to determine the frequency of each segment $s_i$ according to its $P(s_i)$ value. Low frequency for highly demanded segments (those in Part 2) increases access latency $A$, whereas an unnecessarily high frequency increases the length of the program and $A$. Thus, the segments' frequencies have to be optimized. As mentioned, for the retrieval of segment $s_i$, we have to retrieve all segments $s_j$ for $0 \leq j < i$ as well. Since the segments $s_j$ in Part 1 have small $P(s_j)$ values, if we assign them a low frequency, then they will become the "weakest link in the chain" and will increase $A$. Considering all the aforementioned issues, we describe the following problem that has to be solved.

**Problem 1** *Given a sequence $S = \langle s_0, \ldots, s_{N-1} \rangle$ of $N$ segments*[5] *and their access probabilities $P(s_i)$, $0 \leq i \leq N$, we want to find: (1) a $k$ value, $0 \leq k < N$, which determines a subsequence of consecutive segments $G = \langle s_0 \ldots s_{k-1} \rangle$, and (2) a frequency value $f \geq 1$ for the segments in $G$. The values of $k$ and $f$ are selected such that the corresponding program minimizes the expected access latency $A$.*

That is, we divide $S$ in two subsequences: $G$ and $G' = S - G$. Intuitively, $G$ contains the highly requested segments and their preceding ones (i.e., those in Parts 1 and 2), to which we assign frequency $f \geq 1$.[6] To the segments of $G'$ we assign frequency equal to one. This means that: (i) these segments are not repeated within a cycle and (ii) the frequency of segments in $G$ (those that are going to be repeated) will be defined relative to the basic frequency of segments in $G'$. In the following, we present the proposed scheduling method by describing methods to generate non-flat programs and to derive the values for $k$ and $f$.

# 4 Proposed scheduling method

## 4.1 Program generation

Assume (for now) that the values of $k$ and $f$ are given. A non-flat program can be generated by first placing the segments of $G$ at $f$ distinct positions in the program and next by filling the remaining positions with the segments from $G'$. In general, between any two consecutive appearances of segments from $G$, the number of segments from $G'$ may vary. An example is depicted in Figure 4a (assuming $k = 2$). In contrast, the number of intermediate segments from $G'$ can be fixed. An example is depicted in Figure 4b. According to [1], the former case represents a *skewed* program, whereas the latter a *non-skewed* program. Henceforth, we are based on non-skewed programs, because they result in smaller latency compared to skewed ones [1]. Given $k$ and $f$, it follows that the length of a non-skewed program (in number of segments) is equal to $(f-1)k+N$, whereas the number of intermediate segments between two appearances of segments from $G$, is equal to $\lceil \frac{N-k}{f} \rceil$.

## 4.2 Computing parameters for access-latency minimization

In this section we describe the computation of $k$ and $f$. Let $A(s_i)$ be the access latency (in terms of number of segments) for a user that request the segments from $s_0$ up to $s_i$. The expected access latency for all segments is, therefore, equal to:

$$A = \sum_{i=0}^{N-1} P(s_i) A(s_i). \quad (1)$$

$A(s_i)$ depends on whether $i < k$ or not. For this reason we consider the following:

**Lemma 1** *For a segment $s_i$, $0 \leq i < N$, it holds that:*

$$A(s_i) = \begin{cases} \frac{1}{2}\left(\lceil \frac{N-k}{f} \rceil + k + i + 1\right), & 0 \leq i < k \\ \frac{1}{2}\left(\lfloor \frac{i-k+1}{\lceil \frac{N-k}{f} \rceil} \rfloor k + fk - 3k + N + i + 1\right), & k \leq i < N. \end{cases}$$

*Proof.* For $0 \leq i < k$ we consider the following two cases.

*Best case*: If the user issues the request at the moment that $s_0$ is about to be broadcast, then at best case we get that $A_b(s_i) = i+1$, because the user has to wait for the retrieval of the segments from $s_0$ to $s_i$.

*Worst case*: If the user issues the request at the moment that $s_i$ has just been broadcast, then has first to wait for the $k-i-1$ remaining segments from $G$ to be transmitted (that is, $s_{i+1}$ to $s_{k-1}$), next for $\lceil \frac{N-k}{f} \rceil$ intermediate segments from $G'$, and finally $i+1$ segments from $G$ (that is, $s_0$ to $s_i$). Thus, at worst case, we get $A_w(s_i) = \lceil \frac{N-k}{f} \rceil + k$. On average we have that $A(s_i) = \frac{1}{2}(A_b(s_i) + A_w(s_i))$. With simple algebraic manipulations we get the required equality for the case $0 \leq i < k$.

For $k \leq i < N$ we consider again the following two cases.

[5] For JPEG, $N = 64$.

[6] When $f = 1$, a flat program is produced.


(a) (b)

**Figure 4. Example of two programs: (a) skewed; (b) non skewed.**

*Best case*: If the user issues the request at the moment that $s_k$ (the first segment in $G'$) is about to be broadcast, then has to wait for $i-k+1$ segments from $G'$. Additionally, we have to wait for all the appearances of the segments of $G$ between $s_0$ and $s_i$. Since for each $\lceil \frac{N-k}{f} \rceil$ segments from $G'$ there are $k$ segments from $G$ and we have $i-k+1$ segments from $G'$, we have to additionally wait for $\lfloor \frac{i-k+1}{\lceil \frac{N-k}{f} \rceil} \rfloor k - k$ segments of $G$.[7] Therefore, $A_b(s_i) = \lfloor \frac{i-k+1}{\lceil \frac{N-k}{f} \rceil} \rfloor k - k + i - k + 1$.

*Worst case*: If the request is issued after $s_i$ has just been broadcast, then we have to wait for all the program to be broadcast, that is, $A_w(s_i) = (f-1)k + N$. On average we have that $A(s_i) = \frac{1}{2}(A_b(s_i) + A_w(s_i))$. With simple algebraic manipulations we get the required equality for the case $k \leq i < N$. □

From Equation 1 and Lemma 1 it follows that, for a given $k$, we can consider $A$ as a function of $f$. With the following proposition, we find the value $f^*$ that optimizes $A$.

**Propositon 1** *For a given $k$, $A$ is minimized for the value*

$$f^* = \frac{(N-k)\sqrt{P(G)}}{\sqrt{k\sum_{i=k}^{N-1} iP(s_i) + [k(N-2k+1)]P(G')}} \quad (2)$$

*Proof.* We simplify $A(s_i)$ by not considering the ceiling and floor functions. Next, we take the derivative: $\frac{\partial A}{\partial f} = -\frac{N-k}{2f^2}\sum_{i=0}^{k-1} P(s_i) + \sum_{i=k}^{N-1} \frac{P(s_i)}{2}\left(\frac{i-k+1}{N-k}k + k\right) = 0$. We denote $P(G) = \sum_{i=0}^{k-1} P(s_i)$ and $P(G') = \sum_{i=k}^{N-1} P(s_i)$. Therefore, by solving for $f$ and with simple algebraic manipulations, we get the required equation. Because the frequency is an integer, we have to take the ceiling $\lceil f^* \rceil$ value. □

Finally, we derive the optimum $k^*$ value for $k$ as following:

$$k^* = \arg \min_{0 \leq k < N} A, \quad (3)$$

where we assume that for each examined $k$, we determine the optimum $f^*$ according to Equation 2.

[7] The term $(-k)$ results because, in the best case, the request is issued at the broadcast of $s_k$, not of $s_0$.

The time cost to determine $k^*$ is $O(N^2)$, due to the $N$ possible values in Equation 3 and the summation performed in the denominator of Equation 2 for each possible value.

### 4.3 Tuning time

The proposed scheduling minimizes access latency. However, wireless broadcasting also involves *tuning time*, that is, the time a mobile client remains tuned in the broadcast channel. Tuning time is related to client's battery consumption, thus it has to be minimized too. We consider the following method. Each bucket in a segment contains, as associated metadata, the following:

- the order of its coefficients,
- the position $p$ of the corresponding segment in the program, i.e., the offset of the segment from the first segment in the program (e.g., in Figure 4b, the position $p$ of segment $s_5$, is 7),
- the segment's size (in buckets), and
- the values of $N$, $f$, and $k$.

These metadata produce negligible space overhead within each bucket. Upon its tuning to the channel, a client reads the metadata and computes the *relative position* $r$ of the segment, which is given as follows:

$$r = p - \lfloor \frac{p}{c} \rfloor c$$

where $c = k + \lceil \frac{N-k}{f} \rceil$. The relative position $r$ of a segment denotes its offset from the first segment of the most previous repetition of segments in $G$ (the ones with frequency higher than one). In the example of Figure 4b, the relative position $r$ for segment $s_5$, is equal to 3, because $s_5$ is three segments away from the most previous repetition of segment $s_0$ (in this case, $G$ consists of $s_0$ and $s_1$).

After reading the metadata and computing $r$, the client disconnects to reduce energy consumption. While disconnected, the client knows each following position $p$ (because

transmission rate is constant). For a segment $s_i$ in each forthcoming position $p$, the client can calculate the order $i$ of its coefficients, using the following function:

$$i = \begin{cases} r, & r < k \\ r - k + \lfloor \frac{p}{c} \rfloor \lceil \frac{N-k}{f} \rceil, & otherwise. \end{cases}$$

Thus, when the currently broadcast segment is in the range required by the user, i.e., when $0 \leq i \leq m$, the client tunes and reads it. This minimizes tuning time, as the client tunes only to read the requested segments. Experimental results in the following section will demonstrate the efficiency of this technique.

## 5 Experimental results

The performance of the proposed scheduling method, henceforth denoted as *Image Broadcast Scheduling* (IBS), was tested experimentally. We compared IBS against the flat program generation and MAD [6], one of the best scheduling algorithms for single-item retrieval. In order to control the various characteristics of the data, we used a simulation environment, where all scheduling algorithms were programmed in C language. $P(s_i)$ values where synthetically generated following the Poisson distribution. The main parameter was the *peak* point of the distribution, which is equal is equal to $\lambda$ variable (the peak point is given relatively to the total number of segments $N$). The peak point is corresponds to the approximation level that is requested by the majority of clients. The shapes of the generated probability distributions were similar to that of Figure 3.

In our first experiment we measured the $f^*$ and $k^*$ values produced by IBS. The results are depicted in Figures 5 and 6, respectively. As expected from Equations 2 and 3, $f^*$ and $k^*$ increase with increasing peak value.

In the following experiment we measured the access latency, $A$, resulted by the program of IBS and by the flat program. $A$ is measured versus the peak point of $P(s_i)$. The results are depicted in Figure 7. In the same figure we also plotted the value of $A$ that is predicted by Equation 1 (denoted as Pred IBS), to measure the accuracy of the prediction. For IBS, the actual and predicted values are very similar. Moreover, IBS clearly outperforms the flat program. With increasing peak value, the latency of both methods converges to a point, as clients retrieve most of the segments.

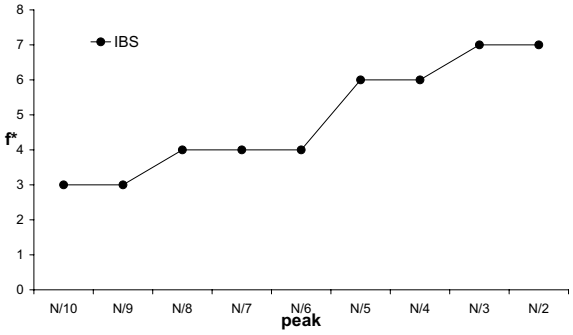



**Figure 5. Resulting $f^*$ vs. peak.**

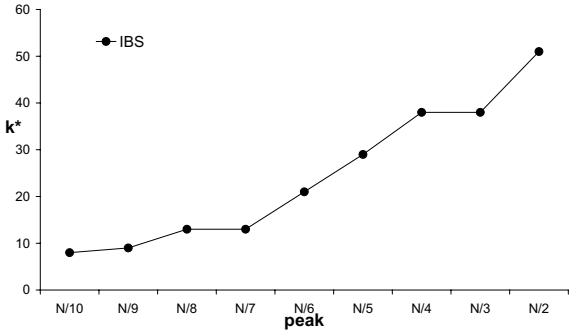



**Figure 6. Resulting $k^*$ vs. peak.**

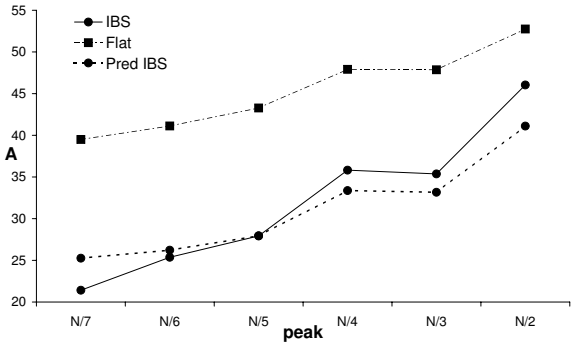



**Figure 7. Comparison between IBS and Flat.**

Figure 8 illustrates the results of the comparison between IBS and MAD. IBS compares favorably against MAD, as the latter is designed for single-item retrieval, thus favors some particular segments without considering the correlations between the different segments. This unnecessarily increases the length of the program and, thus, $A$.

Next, we examined how fast each method retrieves the most *important* segments. The importance of a segment $s_i$ stems from the fact that $s_i$ is needed to retrieve all $s_j$ with $j > i$; thus these $s_j$ segments depend on $s_i$. To quan-

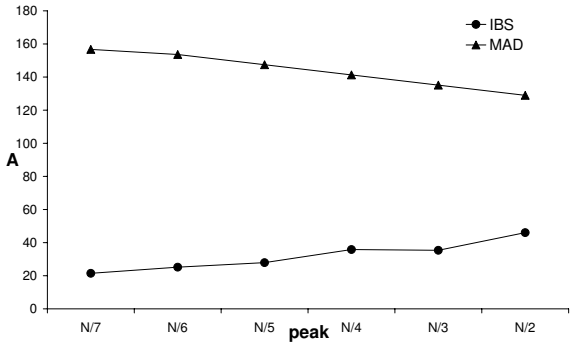



**Figure 8. Comparison between IBS and MAD.**

tify the importance $I(s_i)$ of a segment $s_i$, we consider that $I(s_i) = \sum_{j \geq i}^{N} P(s_j)$. Intuitively, a segment is more important, when the access probability of all its dependent segments is high. For a given collection $R$ of retrieved segments, the total importance, $I$, is the sum of $I(s_i)$ for all $s_i \in R$. ($I(s_i)$ is considered only once for each retrieved $s_i$.) Figure 9, depicts the resulting $I$ for all algorithms against the size of collection $R$, given relatively to the total number of segments $N$. When this size is small, MAD presents the best performance, as it tends to favor the segments with high probability of retrieval. However, as explained, this happens at the cost of very high $A$ that MAD incurs. IBS presents very good performance, whereas in several cases it is better than MAD. The flat program clearly looses out, as it treats all segments uniformly.

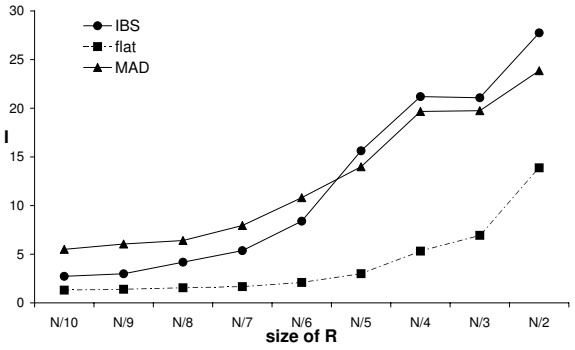



**Figure 9. Results on importance $I$.**

We also tested the tuning times, in terms of number of buckets, for all the examined algorithms. The results are depicted in Figure 10 (the vertical axis is in logarithmic scale). As expected, MAD has the worst tuning time, because it results to programs where segments with consecutive order of coefficients are very scattered. Thus, the client has to stay tuned for long times in order to retrieve the required range of segments. In contrast, the flat program is expected to have the optimum tuning time, because the client can tune only when the required segments are being transmitted. IBS, due to the technique described in Section 4.3, is able to have tuning time similar to the optimum (in Figure 10 the curves for IBS and Flat are overlapped). This indicates the efficiency of IBS not only with respect to reduced access latency, but to the minimization of tuning time as well.

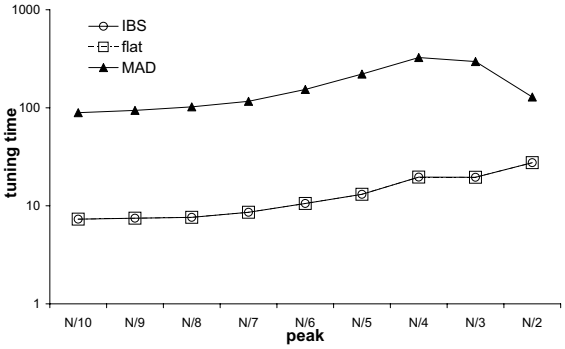



**Figure 10. Results for tuning time.**

## 6 Conclusions

Due to the advances in wireless technology, image broadcasting is expected to find wide acceptance in wireless broadcast networks (WBNs) in the near future. Since in several real-world applications, mobile clients have overlapping interests about the same visual information (e.g., using a PDA to view maps or images of headline news in a browser), it is more efficient to broadcast the requested images than simply to unicast them separately to each client.

In this paper, we have examined the problem of image broadcasting in WBNs. We proposed a novel scheduling method for non-flat program generation. It is designed based on characteristics of existing standards for image encoding. We examined the problem of minimizing the time latency that results when requesting an image from a WBN. We also considered the issue of energy consumption by minimizing tuning-time. Experimental results verified the superiority of the proposed method for minimizing access latency and tuning time. Therefore, we propose an efficient and simple to implement method for the problem of image broadcasting.

Future work will include extensions to multiplex many images in the broadcast program and the examination of multi-channel networks.

## References


[1] S. Acharya, R. Alonso, M. J. Franklin, and S. B. Zdonik. Broadcast disks: Data management for asymmetric communications environments. In *Proc. ACM Conf. on Management of Data (SIGMOD'95)*, pages 199–210, 1995.

[2] A. Bar-Noy, R. Bhatia, J. Naor, and B. Schieber. Minimizing service and operation costs of periodic scheduling. In *Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA'98)*, pages 11–20, 1998.

[3] G. Lee, S.-C. Lo, and A. L. P. Chen. Data allocation on wireless broadcast channels for efficient query processing. *IEEE Transactions on Computers*, 51(10):1237–1252, 2002.

[4] G. Pekhteryev, Z. Sahinoglu, P. Orlik, and G. Bhatti. Image transmission over IEEE 802.15.4 and ZigBee networks. In *Proc. IEEE Int. Symposium on Circuits and Systems (ISCAS)*, 2005.

[5] K. Stathatos, N. Roussopoulos, and J. S. Baras. Adaptive data broadcast in hybrid networks. In *Proc. Conf. on Very Large Databases (VLDB'97)*, pages 326–335, 1997.

[6] C. J. Su, L. Tassiulas, and V. Tsotras. Broadcast scheduling for information distribution. *Wireless Networks*, 5(2):137–147, 1999.

[7] K.-L. Tan and J. X. Yu. Generating broadcast programs that support range queries. *IEEE Transactions on Knowledge and Data Engineering*, 10(4):668–672, 1998.

[8] N. Thomos, N. V. Boulgouris, and M. G. Strintzis. Wireless image transmission using turbo codes and optimal unequal error protection. *IEEE Transactions on Image Processing*, 14(11):1890–1901, 2005.

[9] Liberatore V. Multicast scheduling for list requests. In *Proc. IEEE Joint Conf. of Computer and Communications Societies (INFOCOM'02)*, 2002.

[10] G.K. Wallace. The jpeg still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1):xviii–xxxiv, 1992.

[11] W. Yu, Z. Sahinoglu, and A. Vetro. Energy efficient JPEG 2000 image transmission over wireless sensor networks. In *Proc. IEEE Global Telecommunications Conference (GLOBECOM), Vol. 5*, pages 2738–2743, 2004.