A Data Allocation Method over Multiple Wireless Broadcast Channels

SUNGWON JUNG1 AND SOOYONG PARK2
1,2Department, of Computer Science and Engineering
Sogang University
1 Shinsoo-Dong, Mapo-Gu Seoul, 121-74, Korea

In this paper, we concentrate on data allocation methods for multiple wireless broadcast channels to reduce the average data access time. Existing works first sorted data by their access probabilities and allocate the partitions of the sorted data to the multiple wireless channels. They employ the Flat broadcast schedule for each channel to cyclically broadcast all the data items allocated to it. The different access probabilities of the data items within a channel are ignored. To cope with this problem, S2AP method was proposed. It allocates a popular data item more than once per cycle to the channel to which it is assigned. The number of times that each data item is allocated reflects its access probability. However, the performance improvement of S2AP method is somewhat limited because the skewness of data access probability distribution within each channel is not large. We propose ZGMD method which first allocates data over multiple wireless channels by trying to maximize the average skewness of data access probability distributions over multiple channels. ZGMD method then computes the broadcast repetition frequencies of all the data items in each channel by using the method proposed in S2AP scheme. Finally, ZGMD method generates the broadcast disk program for multiple wireless broadcast channels. Our performance analysis shows that ZGMD method gives the better average access time than the existing methods.

Keywords: Broadcast Channels, Data Allocation, Mobile Clients, Mobile Databases

1. INTRODUCTION

The use of laptops, portable computers, and PDAs over mobile communication networks is increasing, and mobile applications such as monitoring stock market prices, traffic reports, and weather forecasts are widely available in wireless environments. In such an environment, a large number of mobile clients may be querying databases over unreliable, slow, and expensive wireless communication links. Data broadcasting is an efficient method of data dissemination that can overcome the limitations of a wireless environment, such as its low communication bandwidth and the energy constraints of mobile devices. A data server can broadcast data periodically, and mobile clients can listen on one or more channels to access the data that they require.

In a broadcast environment, mobile clients need to wait for the data that they want to appear on a channel. The time spent waiting for a data item is called its expected delay. Clearly, as the number of data items to be broadcast increases, the average expected delay increases. In a multichannel environment, data that is requested more frequently can
be sent over a fast channel, which carries less data and has a short broadcast cycle length. Less popular data items can be allocated to a slow channel with a longer broadcast cycle length. Organizing data to make use of multiple broadcast channels raises a number of interesting challenges. One is the algorithm that partitions data among the channels. The server needs to generate a broadcast schedule for multiple channels in a way that reduces the average expected delay of the data items. The design of such a scheduling algorithm has already attracted significant research [1-11].

However, these works have been focused on the assignment of a data item to a channel and does not respect the different access probabilities of data items in each channel. To cope with this problem, S2AP method is proposed [12]. It allocates a popular data item more than once per cycle to the channel to which it is assigned. The number of times that each data item is allocated reflects its access probability. However, the performance improvement of S2AP method is somewhat limited. This is because S2AP method still uses the same data partition scheme as the existing works do. That is, all the existing works including S2AP method first sort data items by their access probabilities and allocate the partitions of the sorted data to the multiple wireless broadcast channels. In other words, each partition of sorted data is horizontally allocated to a single channel. As a result, the skewness of data access probability distribution in each channel becomes small. This small skewness of data access probability distribution makes S2AP method generate the less effective broadcast disk program for multiple wireless broadcast channels, which in turn increases the average data access time of mobile clients.

In this paper, we propose ZGMD(Zipfian Generated Multichannel Data allocation) method which allocates data over multiple wireless channels by trying to maximize the average skewness of data access probability distributions for multiple wireless channels. In ZGMD method, we first sort data items by their access probabilities and allocate the sorted data vertically across the multiple wireless channels such that the average skewness of data access probability distributions over multiple channels is maximized. We then computes the broadcast repetition frequencies of all data in each channel by using the same method used in S2AP and generates the broadcast disk program for multiple wireless broadcast channels.

The rest of this paper is organized as follows. In section 2, we overview related work and the preliminaries required to characterize the allocation problem. In Section 3, we propose ZGMD method which allocates data items to multiple wireless channels by maximizing the skewness of data access probability distributions of all channels. An example showing broadcast schedules produced by ZGMD method is also presented in Section 3. Our scheme has been evaluated through experimental simulations, which are reported in Section 4. Finally, Section 5 provides the concluding remarks.

2. RELATED WORKS

Past works on finding optimal allocation numbers for broadcast data are based mostly on single-channel environments [13, 14, 15]. However, the optimal allocation numbers on a single channel may not remain optimal when the number of channels is increased and data items are split over multiple channels. Several algorithms have been proposed for the data allocation problem over multiple channels. In this section, we will compare five
These schemes: Flat[1], Bin–Packing[2], VF [3], Greedy[4,16], and S2AP[12]. Table 2.1 shows the main characteristics of these algorithms.

<table>
<thead>
<tr>
<th>Data Allocation Schemes</th>
<th>Basic Allocation Concept</th>
<th>Complexity</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flat [1]</td>
<td>$\frac{N}{K}$</td>
<td>$o(K)$</td>
<td>Simple, low computational overhead, long data access time for skewed access pattern</td>
</tr>
<tr>
<td>Bin–Packing[2]</td>
<td>Based on bin-packing. Allocates a data item to the next channel(bin) if the current bin temperature exceeds the ave. bin temperature</td>
<td>$o(N \log N)$</td>
<td>Simple, data temperature is considered, needs to run cooling process periodically</td>
</tr>
<tr>
<td>VF $^K$ [3]</td>
<td>Heuristic method to compose the allocation tree which minimize the level costs</td>
<td>$o(KN^2 \log K)$</td>
<td>Average expected delay is close to the optimal schedule, high computational complexity and space to maintain cost table</td>
</tr>
<tr>
<td>Greedy [4]</td>
<td>Greedy algorithm to determine split-points which minimize the average expected delay</td>
<td>$o((N+K)\log K)$</td>
<td>Average expected delay is close to the optimal schedule, relatively low computational overhead</td>
</tr>
<tr>
<td>S2AP[12]</td>
<td>Greedy algorithm to determine split points &amp; repetition frequencies of data which minimizes the ave. expected delay</td>
<td>$o(KN^2)$</td>
<td>Higher computational complexity than Greedy method [4] but its performance is better</td>
</tr>
</tbody>
</table>

These algorithms all use the following system model: There are $K$ broadcasting channels, each of which is denoted by $C_i$, where $1 \leq i \leq K$. A server broadcasts $N$ data items to mobile clients over the $K$ channels. Each data item $d_i$ is of the same size and it has its own access probability $p_i$ where $1 \leq j \leq N$ and $\sum_{j=1}^{N} p_j = 1$. We assume that the access probability $p_j$ of data item $d_j$ is always greater or equal to the access probability $p_i$ of data item $d_i$ for $1 \leq i \leq j \leq N$. In other words, data items $d_1, d_2, ..., d_N$ are in the descending order of their access probability $p_1, p_2, ..., p_N$. Each channel $C_i$ broadcasts $N_i$ data items, and the total number of data items is $\sum_{i=1}^{K} N_i = N$. Each channel broadcasts its data items cyclically. Time is slotted into units called ticks, which are defined as the amount of time necessary to transmit a unit of data. A mobile client requests a single item at a time. Requests occur independently at a constant average rate, and $p_j$ does not vary from tick to tick. It is also assumed that clients know a priori the contents of the channels. This simplifying assumption is made because the indexing problem is orthogonal to the channel allocation problem: We may assume that there is an index channel which supplies data channel information to clients. For the rest of this paper, we will also use this system model to explain our proposed ZGMD method.

The performance of the data allocation schemes itemized in Table 2.1 was measured in terms of average expected delay (i.e., average access time). Note that access time and expected delay has the same meaning and they are used interchangeably. The expected delay, denoted by $w_j$, is the expected number of ticks (number of data items to be skipped) that a client must wait for the broadcast of data item $d_i$. When $N$ items are
cyclically broadcast on a single channel, \( w_i \) become \( N/2 \). The single-channel average expected delay (SCAED) is the average number of ticks that a client must wait for a request and is computed as the sum of all expected delays, weighted by their access probabilities. Thus, SCAED is defined as follows:

\[
\text{SCAED} = \sum_{i=1}^{N} w_i p_i = \frac{N}{2} \sum_{i=1}^{\xi} p_i
\]

When \( N_j \) items are broadcast cyclically on channel \( C_i \), the average expected delay in receiving a data item \( d_j \) on \( C_i \) is \( N_j / 2 \). Hence, multichannel average expected delay (MCAED) is defined as follows:

\[
\text{MCAED} = \frac{1}{2} \sum_{i=1}^{N} (N_j \sum_{d_j \in C_i} p_j)
\]

This equation was used to measure the performance of the allocation schemes for multiple wireless broadcast channels.

The Flat scheme simply allocates an equal number of data items to each channel. It can work quite efficiently if all the data items have approximately equal access probabilities. However, this does not reflect the usual situation in practice, and a server needs to respect the difference in the access probabilities of data items. This is why skewed allocation methods such as Bin-Packing, VF\( k \), and Greedy were proposed. Bin-Packing creates a broadcast schedule, which nearly equalizes the total access probability of each channel. VF\( k \) uses a heuristics to construct an allocation tree with variant fan-out. The Greedy method also uses a heuristic algorithm to determine split-points for different channels (partitions).

As we mentioned earlier, these schemes aim to schedule broadcasts by finding points at which to split data between channels so as to minimize the average expected delay. However, all the data items allocated to the same channel will share the same expected delay, even though they all have different access probabilities. To cope with this problem, S2AP (Sensitive to Access Probability) method is proposed. It allocates some data items several times to a single channel. First, allocation numbers are determined, which are the number of times that each data item will be allocated in a major broadcast cycle. Then, each data item is assigned to one of the channels. Finally, the items are allocated to their assigned channels the number of times specified by their allocation numbers.

S2AP method proposed two new channel average expected delays called S2AP-SCAED and S2AP-MCAED. The S2AP-SCAED is derived from the SCAED. S2AP-MCAED, which is derived from the MCAED, is the multiple channel version of S2AP-SCAED. Assume that data items are sorted in descending order by access probability \( p_j \), and that data items of uniform size are allocated to channels uniformly. A data item \( d_j \) is allocated \( t_j \) times in a major broadcast cycle. The total length of a major broadcast cycle over a single channel \( C_i \) is \( N_j = \sum_{d_j \in C_i} t_j \). Then, S2AP-SCAED is the average expected delay when a data item \( d_j \) is allocated \( t_j \) times over a single channel \( C_i \) and is expressed as:

\[
\text{S2AP-SCAED} = \frac{N'}{2} \sum_{d_j \in C_i} p_j t_j
\]

S2AP-MCAED is the average expected delay when a data item \( d_j \) is allocated \( t_j \) times over multiple channels \( C_i \), where \( 1 \leq j \leq N \) and \( 1 \leq i \leq K \). It is formally expressed as:
In S2AP method, it first sorts data items by their access probabilities, and tries to find the optimal partitions of the sorted data by using greedy algorithm to allocate them to the multiple wireless broadcast channels. The data allocation based on greedy algorithm creates hot and cold channels. Hot and cold channels consist of the data items with high and low data access probabilities respectively. This kind of data allocation usually gives the small data access probability differences between the successive data allocated within each channel. As a result, S2AP method is likely to give the small variation of \( t_j \) of the data \( d_j \). This creates a problem for S2AP method since it generates the less effective broadcast disk program for multiple wireless broadcast channels, which in turn increases the average data access time of mobile clients. This kind of problem will become severe as the total number of multiple wireless broadcast channels increase.

To cope with the problem of S2AP method, we propose ZGMD method which allocates data over multiple wireless channels by trying to maximize the average skewness of data access probability distributions for multiple wireless channels. The increased skewness of data access probability distribution for each channel gives the large data access probability differences between the successive data allocated within each channel. As a result, S2AP method is likely to give the large variation of \( t_j \) of data \( d_j \). For the rest of our paper, we now call the allocation time \( t_j \) of data \( d_j \) as the repetition frequency of data \( d_j \). In the next section, we discuss our ZGMD method.

3. ZGMD METHOD

In this section, we propose ZGMD (Zipfian Generated Multichannel Data allocation) method that generates a broadcast program for multiple broadcast channels. ZGMD consists of three steps. In the first step, ZGMD allocates data to multiple channels so that the average skewness of data access probability distributions of all channels is maximized. Next, for the data allocated in each channel, ZGMD computes their broadcast frequencies. At the last step, by using the broadcast frequencies of data, ZGMD method generates a broadcast disk program for each channel. We first describe how to measure the average skewness of data access probability distribution for a single channel and then for the multiple channels.

3.1 Measurement of Skewness of Data Access Probability Distribution

We define a set of data access probabilities \( p_i \) of data \( d_i \) allocated in a channel \( C_j \) as the data access probability distribution for the channel \( C_j \). To measure the skewness of data access probability distribution for a single channel \( C_j \), we use zipfian factor \( \theta_j \) in the Zipf’s law expressed by the equation (5). In the equation (5), \( f_{j,i} \) represents the access frequency of data \( d_i \) allocated to a channel \( C_j \). \( r_{j,i} \) represents the descending order rank of the access probability \( p_i \) of data \( d_i \) among the data items allocated to the channel \( C_j \). Note that \( G_j \) is a given constant in Zipf’s law for the channel \( C_j \).
Let $F_{ij}$ be the sum of all the access frequencies of data items allocated to the channel $C_j$. We can formally define $F_{ij} = \sum_{i=1}^{N_j} f_{ji}$. The access probability $P_{ji}$ of a data item $d_i$ allocated to the channel $C_j$ can be defined as $p_{ji} = f_{ji} / F_{ij}$. We replace $f_{ji}$ in the above equation (5) by $p_{ji} = f_{ji} / F_{ij}$. Then, we can represent the access probability $P_{ji}$ of a data item $d_i$ allocated to the channel $C_j$ as follows:

$$p_{ji} = \frac{G_j}{(r_{ji})^{\theta_j} \times F_{ij}}$$

(6)

As the zipfian factor $\theta_j$ increases, the skewness of data access probability distribution for a single channel $C_j$ also increases. Given the data access probabilities $P_{ji}$, by using the equation (6), we want to compute the estimation of $\theta_j$ which represents the skewness of data access probability distribution for a single channel $C_j$. Let $\theta_{ji}$ represent the $\theta_j$ which satisfies the equation (6) for data item $d_i$. Then, we can represent the equation (6) as follows:

$$p_{ji} = \frac{G_j}{(r_{ji})^{\theta_{ji}} \times F_{ij}}$$

(7)

Assume that data item $d_w$ is initially allocated to channel $C_j$ such that $r_{jw} = 1$. From the equation (7), we have $p_{jw} = G_j / (r_{ji})^{\theta_{ji}} \times F_{ij}$. Since $r_{jw} = 1$, $p_{jw} = G_j / (r_{ji})^{\theta_{ji}} = G_j / F_{ij}$. From the equation (7), we obtain $\theta_{ji} = \log(G_j / (p_{ji} \times F_{ij})) / \log(r_{ji})$. By using $G_j = p_{jw} \times F_{ij}$, we can further simplify $\theta_{ji} = \log(G_j / (p_{ji} \times F_{ij})) / \log(r_{ji}) = \log(p_{jw} / p_{ji}) / \log(r_{ji})$ for data item $d_i$ where $i > w$ and $d_i \in C_j$. Then, $\overline{\theta_j}$, the estimation of $\theta_j$ for a channel $C_j$, is computed by averaging $\theta_{ji}$ of all the data items $d_i$ allocated in $C_j$. It is expressed as follows:

$$\overline{\theta_j} = \frac{\sum_{i=w+1}^{d_{C_j}} \theta_{ji}}{N_j - 1}$$

(8)

We name $\overline{\theta_j}$ as SCAZD (Single Channel Average Zipf distribution Degree) for a channel $C_j$. Based on SCAZD, we define MCAZD (Multi-Channel Average Zipf Distribution Degree) for $K$ multiple channels in the following equation (9).

$$\text{MCAZD} = \sum_{j=1}^{K} \overline{\theta_j}$$

(9)

Our goal is to allocate data items $d_1, d_2, \ldots, d_N$ to $K$ channels to maximize MCAZD.
3.2 MCAZD Allocation Method

In this section, we propose a greedy algorithm in Algorithm 1 for allocating data to \( K \) channels that increases MCAZD. Algorithm 1 first performs the initial allocation of \( K \) hottest data items \( d_1, d_2, \ldots, d_K \) to the channels \( C_1, C_2, \ldots, C_K \) respectively. This initial data allocation is necessary because the computation of \( \hat{\theta}_j \) is possible only when each channel \( C_j \) gets the allocation of more than one data item. Algorithm 1 then finds the channel \( C_j \) among \( K \) channels for the allocation of the next \( d_{K+1} \) which gives the maximum MCAZD value. It continues to allocate \( d_{K+2}, d_{K+3}, \ldots, d_N \) to \( K \) channels in the same way as it does for \( d_{K+1} \).

Algorithm 1 MCAZD allocation

//INPUT:
// \( d_1, d_2, \ldots, d_N \): a set of \( N \) unit sized data item \( d_i \) sorted by the decreasing order of their access probability
// \( C_1, C_2, \ldots, C_K \): \( K \) number of broadcast channels
// \( p_i \): the access probability of data \( d_i \)
//OUTPUT:
// Allocation of data items \( d_1, d_2, \ldots, d_N \) to \( C_1, C_2, \ldots, C_K \) channels

for \( i=1 \) to \( K \) do
    Allocate data item \( d_i \) to the channel \( C_i \);
end for

for \( i=K+1 \) to \( N \) do
    max = 1;
    for \( j=1 \) to \( K \) do
        //Let \( MD(d_i, C_j) \) be MCAZD when \( d_i \) is allocated to channel \( C_j \);
        Compute \( MD(d_i, C_j) \);
        if \( (j>1 \land (MD(d_i, C_j)>MD(d_i, C_{j-1}))) \) then
            max = \( j \);
        endif
    endfor
    Allocate the data item \( d_i \) to the channel \( C_{max} \);
end for

To demonstrate algorithm 1, we use the eighteen data items shown in Table 3.1 where each data is associated with its access probabilities. In Table 3.1, we denote the data items \( E, P, A, D, N, \ldots, R, Q \) as \( d_1, d_2, \ldots, d_{18} \).

Table 3.1: Data items sorted by their access probabilities

| E  | P  | A  | D  | N  | O  | B  | C  | H  | M  | F  | G  | L  | K  | I  | J  | R  | Q  |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0.25 | 0.18 | 0.14 | 0.10 | 0.09 | 0.06 | 0.06 | 0.06 | 0.05 | 0.04 | 0.04 | 0.03 | 0.02 | 0.02 | 0.01 | 0.01 | 0.01 | 0.01 |

As shown in Figure 3.1, Algorithm 1 initially allocates the first three hottest data items \( E, P, \) and \( A \) in Table 3.1 to the channels \( C_1, C_2, \) and \( C_3 \) respectively. The SCAZDs
of the channel 1, 2, and 3 are set to 0. Algorithm 1 then tries to find the right channel for the allocation of the fourth hottest data \( D \). For this, it first computes three SCAZDs of \( D \) to the channels \( C_1, C_2, \) and \( C_3 \) respectively. The SCAZD for the channel \( C_i \) is computed as:

\[
\overline{\theta}_i = \frac{\theta_i}{N_j} = \frac{\log \frac{P_i}{P_j}}{N_j} = \frac{\log \frac{0.25}{0.1}}{2} = \frac{\log 2.5}{2} = 1.3219.
\]

The SCAZD for the channel \( C_2 \) is computed as:

\[
\overline{\theta}_2 = \frac{\theta_2}{N_j} = \frac{\log \frac{P_2}{P_j}}{N_j} = \frac{\log \frac{1.8}{0.1}}{2} = \frac{\log 18}{2} = 0.8480.
\]

The SCAZD for the channel \( C_3 \) is computed as:

\[
\overline{\theta}_3 = \frac{\theta_3}{N_j} = \frac{\log \frac{P_3}{P_j}}{N_j} = \frac{\log \frac{1.4}{0.1}}{2} = \frac{\log 14}{2} = 0.4854.
\]

By using \( \overline{\theta}_1, \overline{\theta}_2, \) and \( \overline{\theta}_3 \), Algorithm 1 computes the three MCAZDs when \( D \) is allocated to \( C_1, C_2, \) and \( C_3 \). When we allocate \( D \) to the channel \( C_1 \), MCAZD is computed as:

\[
\overline{\theta}_1 = \frac{\overline{\theta}_1}{3} = \frac{1.3219 + 0 + 0}{3} = 0.4406.
\]

Similarly, Algorithm 1 computes three MCAZDs when \( D \) is allocated to \( C_2 \) and \( C_3 \). When we allocate \( D \) to the channel \( C_2 \), MCAZD is computed as:

\[
\overline{\theta}_2 = \frac{\overline{\theta}_2}{3} = \frac{1.3219 + 1.0 + 0}{3} = 0.7740.
\]

When we allocate \( D \) to the channel \( C_3 \), MCAZD is computed as:

\[
\overline{\theta}_3 = \frac{\overline{\theta}_3}{3} = \frac{1.3219 + 0 + 0.6415}{3} = 0.6545.
\]

In the next iteration, Algorithm 1 tries to find the right channel for the allocation of the fifth hottest data \( N \). It computes three SCAZDs \( \overline{\theta}_1, \overline{\theta}_2, \) and \( \overline{\theta}_3 \) when it allocates \( N \) to the channels \( C_1, C_2, \) and \( C_3 \) respectively. The SCAZD for the channel \( C_i \) is computed as:

\[
\overline{\theta}_i = \frac{\theta_i}{N_j} = \frac{\log \frac{P_i}{P_j}}{N_j} = \frac{\log \frac{0.25}{0.09}}{3} = \frac{\log 2.5}{3} = 0.4653.
\]

Similarly, Algorithm 1 computes \( \overline{\theta}_1 = \log 2 / \log 2 = 1.0 \) and \( \overline{\theta}_2 = \log 1.56 / \log 2 = 0.6415 \). Algorithm 1 computes three MCAZDs when \( N \) is allocated to \( C_1, C_2, \) and \( C_3 \). When we allocate \( N \) to the channel \( C_1 \), MCAZD is computed as:

\[
\overline{\theta}_1 = \frac{\overline{\theta}_1}{3} = \frac{1.3219 + 0 + 0}{3} = 0.4406.
\]

When \( N \) is allocated to the channel \( C_2 \), the MCAZD is computed as:

\[
\overline{\theta}_2 = \frac{\overline{\theta}_2}{3} = \frac{1.3219 + 1.0 + 0}{3} = 0.7740.
\]

When \( N \) is allocated to the channel \( C_3 \), the MCAZD is computed as:

\[
\overline{\theta}_3 = \frac{\overline{\theta}_3}{3} = \frac{1.3219 + 0 + 0.6415}{3} = 0.6545.
\]
Figure 3.2(b) shows these three MCAZDs 0.1551, 0.7740, and 0.6545. Since 0.7740 is the maximum of the above three MCAZDs, Algorithm 1 allocates the data item \( D \) to the channel \( C_2 \). We keep doing the above process until all the data items are allocated to one of the three channels \( C_1 \), \( C_2 \), and \( C_3 \). We show the final result of data allocation of MCAZD method in Figure 3.3.

3.3 Generation of Data Broadcast Schedule

In this section, we explain how ZGMD generates data broadcast schedule after MCAZD allocation is done. ZGMD first computes the repetition frequencies of all the data allocated in each channel by using the following Algorithm 2 proposed in S2AP method [12].
After determining the repetition frequencies of all the data items allocated in all channels, ZGMD applies Acharya’s broadcast disk technique [1] to each channel for the generation of its broadcast disk program. Figure 3.4 shows the example of final data broadcast schedule which ZGMD constructed by applying both Algorithm 2 [12] and Acharya’s broadcast disk technique [1] to the data allocation shown in Figure 3.3. In Figure 3.4, Algorithm 2 computes the repetition frequencies of data items E, D, B, M, L, and J and obtains \( t_E = 6 \), \( t_D = 3 \), \( t_B = 3 \), \( t_M = 2 \), \( t_L = 2 \), and \( t_J = 2 \). We do not explain how Algorithm 2 computes the values of \( t_E \), \( t_D \), \( t_B \), \( t_M \), \( t_L \) and \( t_J \) in this paper, since the detailed demonstration of Algorithm 2 is already given in the example 1 in S2AP method [12].

Then, Acharya’s broadcast disk technique is used to generate a broadcast disk program for channel 1 as follows. First, the data items with the same repetition frequency are grouped together. In this case, we have three groups \{E\}, \{D, B\}, \{M, L, J\}. Next, we compute the LCM(Least Common Multiple) of the repetition frequencies (i.e., 6, 3, 2) of the three groups.

![Figure 3.3: Final result of data allocation in MCAZD allocation algorithm](image)

![Figure 3.4: Final data broadcast schedule of ZGMD method](image)

**Algorithm 2  Repetition frequency determination in S2AP [12]**
"//INPUT:
//u1, u2, ..., uNj: a set of Nj data items allocated to the channel Cj by MCAZD method
//pi: the access probability of data ui where pi ≥ pij for i=1, Nj-1
//start=1, end = Nj, εs = acceptable error

//OUTPUT
//t1, t2, ..., tNj: the repetition frequencies of data items u1, u2, ..., uNj

k = c /*start value to find k*/
while g(k, start, end) < εc, do
    k = k - g(k, start, end) / g(k, start, end) /*Newton-Raphson method*/
end while
for i= start to end do
    ti = Round[pj * (k + pij)] / pj * (k + pij)
end for

function g(k, start, end)
result = 0.0
for i = start to end do
    result = result + pij / (k + pij)
end for
return (result - 1.0)

function g'(k, start, end) /*the derivative of g(k, start, end)*/
result = 0.0
for i = start to end do
    result = result + pij / (k + pij)
end for
return (-result)

Each group is then split into smaller units called chunks. The number of chunks for each
group is computed from dividing LCM by each group's repetition frequency. Finally, a
chunk from each group is scheduled one by one until no chunks are left. This forms a
broadcast disk program for channel 1 as shown in Figure 3.4. Similarly, we can generate
the remaining two broadcast disk programs for the channel 2 and 3, which are also
shown in Figure 3.4.

4. PERFORMANCE ANALYSIS

We analyze and compare the performance of our ZGMD method with Flat[1], VFk[3],
Greedy[4,16], and S2AP[12] methods. We carried out the simulation for the performance
comparison on an Intel Core Duo Processor with 3.0GHz CPU and 4GB memory. To
model the nonuniform (or skewed) data access pattern of mobile clients, we use a Zipf
distribution with a parameter. Note that the Zipf distribution is typically used to model nonuniform access patterns [17]. It produces access patterns that become increasingly skewed as increases. In this simulation, we set the value of $\theta$ to 0.95 to model a skewed access pattern. This access pattern is common in broadcasting environments [17]. The various parameters used in the simulation are tabulated in Table 4.1.

<table>
<thead>
<tr>
<th>Simulation Parameters</th>
<th>Value range</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K$: the number of broadcast channels</td>
<td>1~10</td>
<td>5</td>
</tr>
<tr>
<td>$\theta$: Zipfian parameter</td>
<td>0.0~1.0</td>
<td>0.95</td>
</tr>
<tr>
<td>$N$: the number of broadcast data</td>
<td>500~5000</td>
<td>1000</td>
</tr>
</tbody>
</table>

We choose the average access time of a mobile client as our primary performance metrics. The average access time is measured by averaging the access time of a mobile client over 10,000 simulation runs. The access time (i.e., expected delay) of a mobile client represents the amount of time that the mobile client must wait for accessing the broadcast of a single data item. Note that we measure the access time in terms of broadcast units. A broadcast unit represents an amount of time taken to broadcast a single data item. In the following section, we analyze the performance of Flat, VFK, Greedy, S2AP, ZGMD methods over various parameter values.

### 4.1 Effects of the Number of Broadcast Data

We varied the number of data items between 500 and 5,000 for $K = 5$ and $\theta = 0.95$. ZGMD performed best, as shown in Fig. 4.1. Compared to Flat, VFK, and Greedy, as the total amount of data grows, the average access times of S2AP and ZGMD increase very slowly.

This result shows the advantage of allocating popular data items multiple times for each channel in S2AP and ZGMD. In particular, ZGMD gives the much smaller average access time than S2AP for a large number of data items. This is because the skewness of data access probability distribution within each channel for ZGMD is much larger than
that for S2AP.

4.2 Effects of the Number of Channels

Fig. 4.2 shows the average access time when $\theta = 0.95$ and $N = 1000$. We varied the number of channels from 1 to 10. The ZGMD outperforms Flat, $VF^k$, Greedy, and S2AP, especially when the number of channels is small. This is because it allocates the data items with high access probabilities many times per cycle. As the number of channels increases, the average access times of all the five methods become similar. When there are enough channels and data is widely distributed, the different data allocation schemes do not affect the performance very much.

For a single channel, ZGMD and S2AP generate the same schedule. The average access times of ZGMD and S2AP is a lot less than those of Flat, $VF^k$, and Greedy. This is because ZGMD and S2AP methods generate the broadcast schedule with the multiple numbers of allocations of hot data items. However, Flat, $VF^k$, and Greedy generate the same flat schedule for a single channel.

![Fig 4.2: Effects of the Number of Channels](image)

4.3 Effects of Varying Access Patterns

Fig. 4.3 shows the average data access time against the Zipfian factor. As the Zipfian factor increases, there are big differences in the data access time for the different allocation methods: Greedy and $VF^k$ allocate more bandwidth to the popular items and show a lot better results than Flat. The average access time of Flat is independent of Zipfian factor as we expected. S2AP gives the smaller average access time than Greedy and $VF^k$ by allocating the popular items many times per broadcast cycle in each channel. ZGMD further achieves reductions in the average access time by maximizing the skewness of data access probability distributions of all channels.
5. CONCLUSION

We have presented a data allocation scheme for multiple wireless broadcast channels that reduces the data access time. Our scheme allocates data items over multiple wireless channels by trying to maximize the skewness of data access probability distribution for each channel. We propose SCAZD (Single Channel Average Zipf distribution Degree) and MCAZD (Multi-Channel Average Zipf Distribution Degree) to measure the skewness of data access probability distribution for a single channel and multiple channels respectively. Like S2AP method, our ZGMD method also gives a higher allocation number to a data item with a higher access probability. Simulation results show that our ZGMD gives the better average access time than all the existing methods.

REFERENCES

Sungwon Jung (鄭盛元) received the BS degree in computer science from Sogang University, Seoul, in 1988 and the MS and PhD degrees in computer science from Michigan State University, East Lansing, in 1990 and 1995, respectively. He is currently a professor in the Department of Computer Science and Engineering, Sogang University. His research interests include mobile databases, mobile computing systems, spatial databases, LBS, and Data Mining.

Sooyong Park (朴守鏞) received the Bachelor of Engineering degree in computer science from Sogang University, Seoul, in 1986, the Master of Science degree in computer science from Florida State University in 1988, and the PhD degree in information technology with a major in software engineering from George Mason University in 1995. He is a professor in the Computer Science Department at Sogang University and the director of the Requirements and Validation Engineering Center that is supported by the Korean Ministry of Knowledge and Economy. During 1996-1998, he served as a senior software engineer at TRW ISC. He is actively involved in academic activities, including as president of the Korean Software Engineering Society, a steering committee member of the Asian-Pacific Software Engineering Conference, and a guest editor of the Communications of the ACM December 2006 issue on software product lines. His research interests include requirements engineering, self-managed software architecture, and software product-line engineering. He is a member of the IEEE and the IEEE Computer Society.