SOM: Dynamic Push-Pull Channel Allocation Framework for Mobile Data Broadcasting

Jiun-Long Huang, Wen-Chih Peng[†] and Ming-Syan Chen, Fellow, IEEE
Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan, ROC
[†]Department of Computer Science and Information Engineering
Nation Chiao-Tung University
Hsinchu, Hsinchu, ROC
E-mail: jlhuang@arbor.ee.ntu.edu.tw, wcpeng@csie.nctu.edu.tw, mschen@cc.ee.ntu.edu.tw

Abstract

In a mobile computing environment, the combined use of broadcast and on-demand channels can utilize the bandwidth effectively for data dissemination. We explore in this paper the problem of dynamic data and channel allocation with the number of communication channels and the number of data items given. We first derive the analytical models of the average access time when the data items are requested through the broadcast and on-demand channels. Then, we transform this problem into a guided search problem. In light of the theoretical properties derived, we devise algorithm SOM to obtain the optimal allocation of data and channels. Algorithm SOM is a composite algorithm which will cooperate with (1) a search strategy and (2) a broadcast program generation algorithm. According to the analytical mode, we devise scheme BIS-Incremental on the basis of algorithm SOM which is able to obtain solutions of high quality efficiently by employing binary interpolation search. In essence, scheme BIS-Incremental is guided to explore the search space with higher likelihood to be the optimal first, thereby leading to an efficient and effective search. It is shown by our simulation results that the solution obtained by scheme BIS-Incremental is of very high quality and is in fact very close to the optimal one. Sensitivity study on several parameters, including the number of data items and the number of communication channels, is conducted. The experimental results shows that scheme BIS-Incremental is of very good scalability which is particularly important for its practical use in a mobile computing environment.

Key words: Data dissemination, dynamic data and channel allocation, mobile computing

1 Introduction

In a mobile computing environment, a mobile user with a power-limited mobile computer can access various information via wireless communication. Applications such as stock activities, traffic reports and weather forecast have become increasingly popular in recent years [29]. It is noted that mobile computers use small batteries for their operations without directly connecting to any power source, and the bandwidth of wireless communication is in general limited. As a result, an important design issue in a mobile system is to conserve the energy and communication bandwidth of a mobile unit while allowing mobile users of the ability to access information from anywhere at anytime [24].

A data delivery architecture in which a server continuously and repeatedly broadcasts data to a client community through a *single* broadcast channel was proposed in [1] in order to conserve the energy and communication bandwidth of a mobile computing system. In a push-based information system, a server generates a broadcast program to broadcast data to mobile users. This broadcast channel is also referred to as a broadcast disk from which mobile users can retrieve data [1][7]. The mobile users need to wait for the data of interest to appear on the broadcast channel. The access time is defined as the time elapsed from the moment a user issues a data request to the point that the requested data item is read [15]. One objective of designing proper data allocation in the broadcast disks is to reduce the average access time of data items. The research issues have attracted a considerable amount of attention, including on-demand broadcast [3][4][5][6], data indexing [9][14][15][17][30][33] and client cache management [4][27][31]. In addition, a significant amount of research effort has been elaborated on developing the index mechanisms [16][20][25] and data allocation algorithms [22][23][28][34][35] in *multiple* broadcast channels. In addition, the bandwidth allocation for multi-cell environments with frequency reuse and inference considered was studied in [32].

In addition to being operated in broadcast mode, channels can be operated in *on-demand mode* (i.e., *unicast mode*) in which a client explicitly sends data requests to retrieve the data items of interest [18][19]. The major advantage of data broadcast is its scalability since the performance of the system does not depend on the number of clients listening to the broadcast channels. However, the performance degrades as the number of data items being broadcast increases. It has been shown that the

combined use of the broadcast and on-demand channels can utilize bandwidth more efficiently for data dissemination [18][19]. Hence, the problem of *dynamic data and channel allocation* is to dynamically partition a given total number of communication channels into broadcasting ones and on-demand ones and to dynamically allocate each data item on broadcast or on-demand channel according to the system workload.

Prior studies of data and channel allocation can be classified into the following three categories: (1) pure on-demand, (2) pure broadcast and (3) dynamic data and channel allocation. The pure on-demand algorithms are used in traditional client/server architectures. All channels are operated in on-demand mode, and all data items are allocated in the on-demand channels. Clients explicitly send data requests to the server to obtain the desired data items. This method is desirable when the number of requests is small and when energy saving is not an issue for the mobile devices. In pure broadcast, all channels are allocated in broadcast mode [1][12][22][35], and all data items are broadcast repeatedly in broadcast channels. This method is useful when the access frequencies of data items are highly skewed (i.e., a small number of data items are of interest to a large group of users).

Dynamic data and channel allocation algorithms are proposed to combine the respective merits of on-demand and broadcast modes and to adapt the change of system parameters including the data access frequencies and the number of users in the system [18][19][26]. In dynamic data and channel allocation, the system dynamically allocates broadcast and on-demand channels in accordance with data requests to achieve optimal data access performance. When the load is heavy, the broadcast channels may significantly relieve the load on on-demand channels by broadcasting frequently accessed data items. When the load is light, on-demand channels can take over to provide instantaneous access to data items.

In this paper, we study the problem of dynamic data and channel allocation. Consider the illustrative example shown in Figure 1. Assume that the data items R_i, $1 \leq i \leq 15$, are of the same size and are sorted by their access frequencies. The number of channels in this example is assumed to be four. In the beginning, two channels are assigned as broadcast channels and the other two are on-demand ones. Five data items are put in broadcast channels and the broadcast program is shown in Figure 1a. When

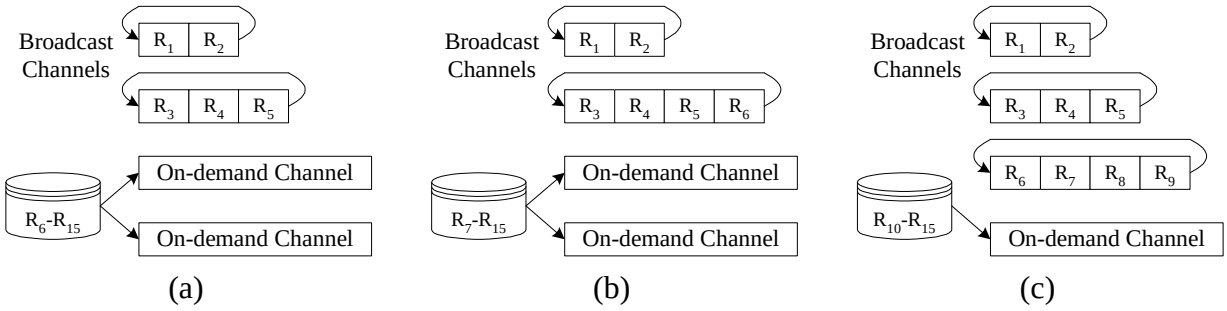


Figure 1: An example scenario of dynamic data and channel allocation

the data request rate increases, R_6 is moved from the on-demand channel to the broadcast channel.[1] This will reduce the data request rate to on-demand channels and the expected waiting time in on-demand channels is hence reduced. The broadcast program is then rescheduled and the new broadcast program is shown in Figure 1b. If the data request rate keeps increasing, as shown in Figure 1c, one channel is re-assigned to be a broadcast one and three data items (R_7, R_8 and R_9) are moved from on-demand channels to broadcast channels. As the partition of broadcast and on-demand channels varies, the number of data items in those channels changes accordingly, showing the dynamic characteristics of this data and channel allocation problem.

We mention in passing that the authors of [26] provide an adaptive algorithm to allocate data items on broadcast and on-demand channels. However, they assume a fixed ratio of the on-demand bandwidth to the broadcast bandwidth. The work in [19] is designed to shuffle the loads among broadcast and on-demand channels to keep the load of on-demand channels in a predetermined region. In [18], the average access time of data items is formulated, and the optimal channel allocation is obtained according to the derived theoretical results. Both works [18] and [19] employed *flat* broadcast programs. A broadcast program is said *flat* if all data items appear with the same frequencies in the broadcast program. On the other hand, a broadcast program is said *hierarchical* if data items of high access frequencies are broadcast more frequently than or equal to those of low access frequencies in the broadcast program. It has been shown that hierarchical broadcast programs usually outperform flat broadcast programs [22][23]. Hence, algorithms proposed by [18] and [19] may not fully utilize network bandwidth. In view of this, we employ *hierarchical* broadcast programs in this paper in order to fully utilize the

[1]The criterion for data movement will be given in Section 4 later.

broadcast channels. This feature distinguishes this paper from others.

Explicitly, we explore in this paper the problem of dynamic data and channel allocation with the number of communication channels and the number of data items given. Gathering the access frequencies of data items is another research issue, since clients do not explicitly send data requests when the data items of interest are put in broadcast channels. Research works [13][36] in gathering or estimating the data access frequencies in broadcast channels can complement our work. Different from the prior studies [18][19], hierarchical broadcast programs are employed in our study. In this paper, we first describe the analytical models of broadcast and on-demand channels and transform the problem of dynamic data and channel allocation into a guided search problem. In light of the theoretical properties derived, we devise five pruning properties which are able to effectively reduce the search space by removing the infeasible solutions from the search space. We then devise algorithm SOM (standing for SOlution Mapping) to obtain the optimal allocation of data and channels. Algorithm SOM is a *composite* algorithm which will cooperate with (1) a search strategy and (2) a broadcast program generation algorithm. According to the analytical models, we devise a search strategy called BIS (standing for Binary Interpolation Search) which is able to dynamically partition the data items and channels into broadcast and on-demand ones in accordance with the incoming requests. Then, based on algorithm SOM, we devise scheme BIS-Incremental to obtain solutions of high quality efficiently by employing BIS as the search strategy and VF^K (standing for Variant-Fanout with the constraint K) as the broadcast program generation algorithm[2]. In essence, scheme BIS-Incremental is guided to explore the search space with higher likelihood to be the optimal first, thereby leading to an efficient and effective search. In addition, scheme BIS-Incremental takes advantage of the incremental property of VF^K which greatly reduces the execution time. It is shown by our simulation results that the solutions obtained by scheme BIS-Incremental are of very high quality and are in fact very close to the optimal ones. Sensitivity study on several parameters, including the number of data items and the number of communication channels, is conducted. Moreover, scheme BIS-Incremental is of very good scalability which is particularly important for its practical use in a mobile computing environment.

The rest of this paper is organized as follows. A description of the related work is given in Section 2.

[2]An introduction of algorithm VF^K will be given in Section 3.1.

In addition, the problem of dynamic data and channel allocation is also formulated. Then the analytical models of broadcast, on-demand channels and the overall system are given in Section 3. In Section 4, we transform the problem of dynamic data and channel allocation into a search problem and develop an efficient algorithm to address this problem based on the derived analytical models. The performance evaluation of the proposed algorithm is presented in Section 5. Finally, this paper concludes with Section 6.

2 Preliminaries

2.1 Related Work

In [2], the architecture consisting of a single uplink channel and a broadcast channel is considered. A portion of time slots on the broadcast channel is allocated to transmit the data items which are explicitly requested by users via the uplink channel. These time slots are said to be in on-demand mode. On the other hand, the remaining time slots are used to transmit all data items according to a hierarchical broadcast program generated by the broadcast disk technique [1]. These time slots are said to be in broadcast mode. In [2], the ratio of the time slots in broadcast mode to those in on-demand mode is fixed, and the broadcast program is static. As a consequence, the scheme proposed in [1] cannot adapt to the change of system workload.

The authors in [26] consider the environment with a broadcast channel, a downlink on-demand channel and an uplink channel. The on-demand channel is dedicated to transmit the data items which are explicitly requested by users via the uplink channel. Flat broadcast programs are employed and only the data items whose request rates are high enough will be allocated on the broadcast channel. The authors propose an algorithm to estimate the popularity of all data items and to dynamically determine the set of data items on the broadcast channel according to the system workload.

In [10], the information system consists of a broadcast channel and an uplink channel. The authors propose an algorithm to prioritize all data items according to the received data requests and the broadcast rates of these data items. Then, the algorithm will allocate the data items with highest priorities on the broadcast channel. The flat broadcast programs are used and $(1,m)$ indexing technique [15]

is employed to construct data indices. The authors also propose several energy efficient data access protocols to minimize the power consumption on data access.

In [19], the authors consider the environments with a single broadcast channel and multiple on-demand channels. The broadcast programs are assumed to be flat. The load of the on-demand channels are first divided into several regions. Then, the authors propose a data allocation algorithm to keep the load of the on-demand channels in a predetermined sub-optimal region by dynamically allocating some data items to the broadcast channel. In addition, the proposed algorithm is able to adaptively adjust the data allocation according to the system workload.

The authors in [18] consider the environments with multiple broadcast and on-demand channels. The broadcast programs on the broadcast channels are assumed to be flat. The authors first model the on-demand channels as an M/M/c queue. Then, the formulae of the average access time of the broadcast and on-demand channels are derived. With these analytical results, the authors propose a data and channel allocation algorithm to determine (1) the numbers of channels which are operated in broadcast and on-demand modes and (2) the data items which are allocated in the broadcast and on-demand channels according to the system workload. However, since the proposed algorithm does not employ hierarchical broadcast programs, the network bandwidth may not be fully utilized. The problem we address is similar to that considered in [18], but different from the latter in that, we also consider the generation of hierarchical broadcast programs to attain a higher network bandwidth utilization.

2.2 System Description and Problem Formulation

Denote the total number of data items as n, and data items as R_i, $1 \leq i \leq n$. Naturally, the n_B frequently accessed data items are placed in broadcast channels and the other $n_O = n - n_B$ data items are in on-demand channels. Let $K = K_B + K_O$ represent the total number of channels where K_B and K_O are the numbers of broadcast and on-demand channels, respectively. The problem of generating broadcast programs for K_B broadcast channels can be viewed as the following discrete minimization problem: Given a set of n_B data items with their access probabilities, partition them into K_B parts so that the average access time of all data items is minimized [12][22][23][35]. Note that once K_B is decided, K_O follows.

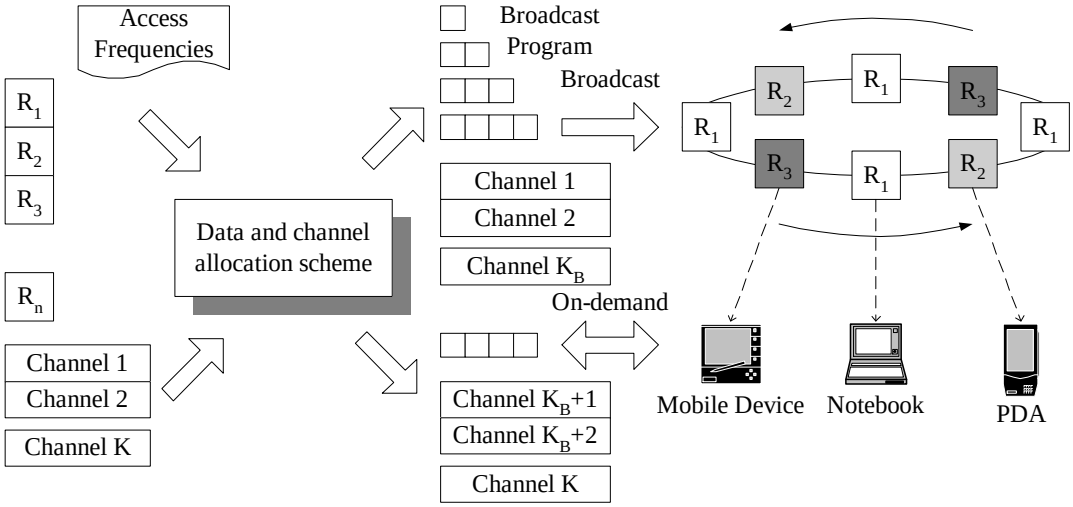


Figure 2: The architecture of a data dissemination system

Figure 2 shows the architecture of a data dissemination system. We assume that each data item is the same size and read-only [18][19]. After being powered on, without knowing the placement of the requested data item, a mobile device has to send a data item request via on-demand channels. If the requested data item is placed in an on-demand channel, the server will reply the data item directly. If the data item is in a broadcast channel, the server replies the broadcast information containing the channel frequencies, the data identifiers, the data index information, and other auxiliary information [18]. After receiving the broadcast information, the mobile device will store the broadcast information in the local storage, listen to the broadcast channel and wait for the requested data item.

If a mobile device already has the broadcast information in its local storage, for each user request, the device will check whether the requested data item is placed in broadcast channels. If yes, the device will tune to the channel where the required data item is placed and wait for the appearance of the requested data item. Otherwise, the device will explicitly send a data request to the server via an on-demand channel and the server will return the requested data item on the on-demand channel.

With the above model, the problem of dynamic data and channel allocation we consider in this paper is formulated as follows:

Problem of dynamic data and channel allocation: Given K channels, n data items and their access frequencies, we shall do the following tasks to minimize the average access time of all data items.

1. Determine the numbers of broadcast and on-demand channels (i.e., K_B and K_O), where $K =$

$K_B + K_O$.

2. Determine the numbers of data items allocated to broadcast and on-demand channels (i.e., n_B and n_O), where $n = n_B + n_O$.

3. Construct a hierarchical broadcast program in the K_B broadcast channels with the n_B most frequently accessed data items.

3 Analytical Models

The analytical models of the broadcast and on-demand channels are given in Section 3.1 and Section 3.2, respectively. In accordance with these analytical models, the overall average access time is formulated in Section 3.3. For better readability, Table 1 lists the symbols used in this paper.

3.1 Broadcast Channels

Since there is more than one data broadcast program for given K_B and n_B, we use $W_B(K_B, n_B)$ to represent the *minimal* average access time of the data items allocated in broadcast channels. Let $C(K_1, n_1)$ be a configuration where $K_B = K_1$ and $n_B = n_1$. The optimal broadcast program can be obtained by executing one broadcast program generation algorithm.

Without considering the use of on-demand channels, the work in [22] explored the problem of generating broadcast programs with the number broadcast channels (i.e., K_B) given. Specifically, the problem of generating broadcast programs for K_B broadcast channels was transformed into a partition problem to partition the data items into K_B partitions. The data items within the same partition are periodically broadcast in the same channel. Two algorithms, OPT and VF^K, were devised in [22] to generate *hierarchical* broadcast programs for multiple broadcast channels. Algorithm OPT is an A*-like algorithm which is able to generate the optimal broadcast program. However, OPT is quiet time-consuming. On the other hand, VF^K is a greedy, heuristic algorithm which is able to efficiently obtain broadcast programs which are shown to be very close to the optimal ones. Since the details of OPT and VF^K are beyond the scope of this paper, interested readers are referred to [22] for the details

Description	Symbol
Number of channels	K
Number of broadcast channels	K_B
Number of on-demand channels	K_O
Number of data items	n
Number of data items in broadcast channels	n_B
Number of data items in on-demand channels	n_O
The j-th data item	R_j
The access frequency of data item R_j	$Pr(R_j)$
The size of each data item	s
The size of each data request	r
The channel bandwidth	b
The data request rate	λ
The average service time for each on-demand channel	$\frac{1}{\mu}$

Table 1: Description of symbols

of OPT and VF^K. To facilitate the design of scheme BIS-Incremental, an overview of VF^K is given as follows.

Basically, VF^K is a partition-based algorithm which divides all data items into K partitions where K is the number of broadcast channels, and allocates all data items into K broadcast channels according to the resultant partitions. Initially, all data items, $R_1, R_2, \cdots R_n$, are reordered according to their access frequencies in descendent order, and are placed in one partition. The average access time of a partition is defined as the average access time of the case that the data items of the partition are broadcast periodically in one broadcast channel. Then, the average access time of a broadcast program on multiple channels is the summation of the average access times of all partitions. In each cut, the partition with the largest average access time, say $\{R_p, R_{p+1}, \cdots, R_q\}$, is selected, and the best cut point of the selected partition, say c, which best reduces the average access time of the broadcast program is determined. Then, the selected partition is cut into two partitions, $\{R_p, R_{p+1}, \cdots, R_c\}$ and $\{R_{c+1}, R_{c+2}, \cdots, R_q\}$. For K_B broadcast channels, $K_B - 1$ cuts are sequentially performed to partition the data items into K_B partitions. Finally, the resultant broadcast program is obtained by periodically broadcasting all data items within the same partition in one broadcast channel.

Then, we have the incremental property of VF^K as follows. For interest of space, the proof of all properties and lemmas is given in Appendix.

Lemma 1 (Incremental Property): The execution of VF^K on configuration $C(K_1, n_1)$ will generate K_1 data broadcast programs of $C(K_b, n_1)$, $1 \leq K_b \leq K_1$.

Lemma 1 means that the execution of VF^K on configuration $C(K_1, n_1)$ will generate K_1 broadcast programs which are the same as the results produced by VF^K for configurations $C(K_B, n_1)$ where K_B =1, 2, 3, $\cdots$, K_1.

3.2 On-demand Channels

Let $W_O(K_O, n_O)$ denote the average access time of the data items placed in on-demand channels. Let $P_O^n(n_O)$ be the probability that the requested data item is in on-demand channels when there are n_O data items placed in on-demand channels. We assume that the arrival process of user requests is a Poisson process with the arrival rate λ. It follows that the arrival process of requests received by on-demand channels is also a Poisson process with arrival rate $\lambda_O = P_O^n(n_O)\lambda$. Same as in [18], we assume that the queueing buffer is infinite. Thus, the on-demand channels are modeled as an M/M/c queueing system [11] with the arrival rate λ_O, the service rate μ and the channel number c. The average service time is $\frac{1}{\mu}$. Let the sizes of data items and data requests be s and r, respectively. Hence, similar to [18], the average service time of on-demand channels can be formulated as:

$$\mu = \frac{b}{s+r}.$$

Omitting the equation manipulation which can be found in [11], the average access time of the on-demand channels (i.e., the M/M/c queueing system where $c = K_O$) when $\rho < 1$ is

$$\text{Average access time} = \frac{1}{\mu} + \left(\frac{r^c}{c!(c\mu)(1-\rho)^2} \right) p_0, \text{ where} \tag{1}$$

$$\rho = \frac{\lambda_O}{c\mu}, r = \frac{\lambda_O}{\mu}, \text{ and } p_0 = \left(\sum_{n=0}^{c-1} \frac{r^n}{n!} + \frac{r^c}{c!(1-\rho)} \right)^{-1}.$$

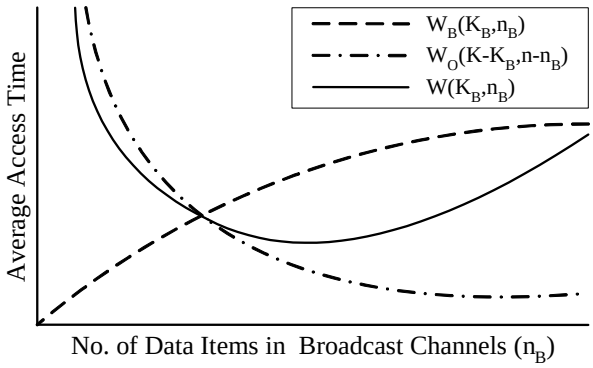


Figure 3: Trade-off for dynamic data dissemination

3.3 Overall Average Access Time

The probability that a user requests a data item placed in the broadcast channels is $P_B^n(n_B) = \sum_{i=1}^{n_B} Pr(R_i)$. On the other hand, the probability that a user requests a data item placed in the on-demand channels is $P_O^n(n_O) = \sum_{i=n-n_O+1}^{n} Pr(R_i) = 1 - \sum_{i=1}^{n_B} Pr(R_i) = 1 - P_B^n(n_B)$. Then, the minimal average access time of a data dissemination system can then be formulated as follows:

$$W_{optimal}(K,n) = \min_{0 \le K_B \le K, 0 \le n_B \le n} \{W(K_B, n_B)\}, \text{ where} \tag{2}$$

$$\begin{aligned} W(K_B,n_B) &= P_B^n(n_B) \times W_B(K_B,n_B) + (P_O^n(n_O)) \times W_O(K_O,n_O) \\ &= P_B^n(n_B) \times W_B(K_B,n_B) + (1 - P_B^n(n_B)) \times W_O(K-K_B, n-n_B). \end{aligned}$$

With K_B predetermined, the relationship among $W(K_B,n_B)$, $W_B(K_B,n_B)$ and $W_O(K-K_B,n-n_B)$ is plotted in Figure 3. Note that $W_O(K-K_B,n-n_B)$ increases exponentially when n_O increases (i.e., when n_B decreases). It is evident that with too few data items in broadcast channels, the volume of requests at the servers may increase beyond their capacity, thereby making the service practically infeasible. On the other hand, the change of the average access time for the broadcast data items is smoother than that for the on-demand data items since the average access time of the broadcast data items only depends on the number of data items allocated to broadcast channels. In this study, the dynamic data and channel allocation algorithm designed will determine the proper values of K_B and n_B with the objective of

minimizing the average access time of all data items.

4 SOM: Solution Mapping on Broadcast and On-demand Channels

In this section, we design algorithm SOM based on the analytical results in Section 3 to address the problem of dynamic data and channel allocation. In Section 4.1, we transform the problem of dynamic data and channel allocation into a search problem and give an overview of algorithm SOM. In Section 4.2, several properties to prune the infeasible solutions from the search space are given. Then, an efficient search strategy based on binary interpolation search, referred to as BIS, is devised in Section 4.3. Based on algorithm SOM, scheme BIS-Incremental, which is able to obtain nearly-optimal solutions by employing BIS and the incremental properties of VF^K, is then proposed. The complexity analysis of scheme BIS-Incremental is given in Section 4.4. Finally, an illustrative example is given in Section 4.5.

4.1 Problem Transformation and Overview of SOM

Given K and n, for each configuration $C(K_B, n_B)$, $W_B(K_B, n_B)$ can be obtained by executing a broadcast program generation algorithm, and $W_O(K - K_B, n - n_B)$ can be calculated by the analytical model of the on-demand channels. As a result, the problem can be transformed into a search problem: to find the configuration with the minimal average access time by searching all given configurations $C(K_B, n_B)$, where $0 \leq K_B \leq K$ and $0 \leq n_B \leq n$.

We design in this section algorithm SOM to address the problem of dynamic data and channel allocation. In essence, algorithm SOM is a composite and generic algorithm which is composed of a search strategy and a broadcast program generation algorithm. Algorithm SOM consists of two major phases: the search space pruning phase and the solution searching phase. Figure 4 shows the architecture of algorithm SOM. In search space pruning phase, some infeasible configurations are removed from the search space. Then, in solution searching phase, a search strategy is used to guide the search

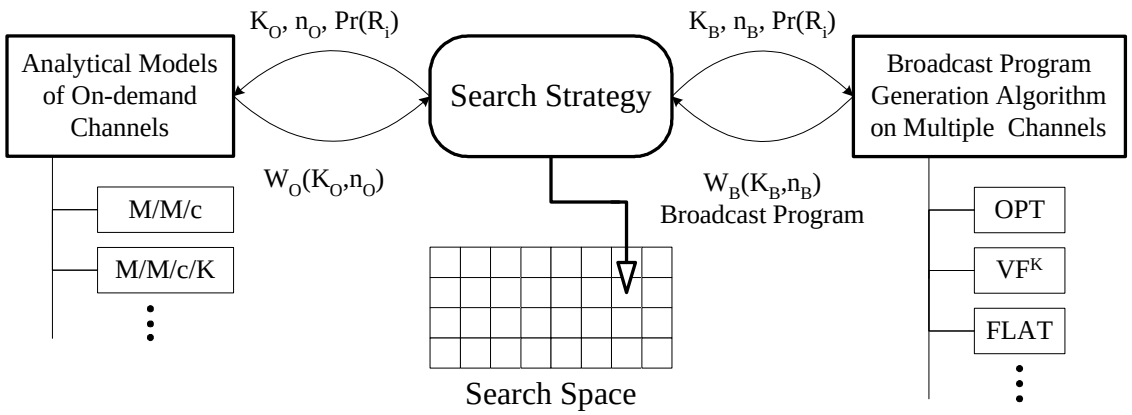


Figure 4: Architecture of algorithm SOM

of the optimal solutions with the aid of the employed broadcast program generation algorithm and the analytical model of the on-demand channels. Note that algorithm SOM does not set any limitation in the broadcast program generation algorithm and the modeling of the on-demand channels. Therefore, any improvement in hierarchical broadcast program generation or on-demand channel modeling can be integrated into algorithm SOM seamlessly.

4.2 Phase One: Search Space Pruning

Initially, the search space should contain all these configurations $C(K_B, n_B)$, where $0 \leq K_B \leq K$ and $0 \leq n_B \leq n$, since they are possible to be the optimal one. Hence, the size of the initial search space is $(K+1) \times (N+1)$. Since on-demand channels are modeled as an M/M/c queueing system, the average access time of the on-demand channels can be derived by Equation (1). Hence, some infeasible configurations can be pruned by the following properties:

Property 1: All configurations that $1 \leq K_B \leq K-1$ and $n_B < K_B$ are pruned since those configurations will not be the optimal.

Analogously, we have the following property.

Property 2: All configurations that $n_B = n$ and $K_B < K$ are pruned, since those configurations will not be the optimal.

Omitting straightforward proofs, we also have the following three properties.

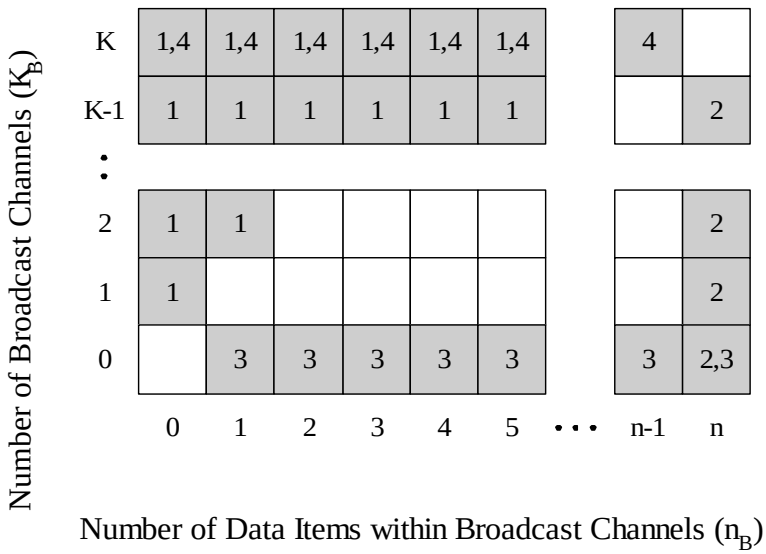


Figure 5: An example of pruned search space

Property 3: All configurations that $K_B = 0$ and $n_B > 0$ are pruned, since if there is no broadcast channel, no data item can be placed in broadcast channels. That is, n_B must be 0 when $K_B = 0$.

Property 4: All configurations that $K_O = 0$ and $n_O > 0$ are pruned, since if there is no on-demand channel, no data item can be placed in on-demand channels. That is, n_O must be 0 when $K_O = 0$.

Property 5: All configurations that $\rho = \frac{\lambda_O}{K_O \mu} \geq 1$ are pruned. When ρ of an M/M/c queueing system is larger than or equal to 1, the system is unstable. That is, the average access time does not converge and will increase drastically as time advances.

Figure 5 shows an example search space where each square represents one configuration. A grey square indicates that this configuration is pruned, and the numbers inside a grey square indicate this configuration is pruned by these properties. Since the number of configurations pruned by Property 5 depends on other parameters such as the request arrival rate, we do not show the configurations pruned by Property 5 in Figure 5.

Lemma 2: When $K \geq 1$ and $n \geq K$, Properties 1-4 are able to prune $2n + \frac{(K-1)(K+2)}{2}$ configurations.

Lemma 3: (1) The lower bound of the ratio of pruned configurations is $\frac{4n+K^2+K-2}{2(n+1)(K+1)}$ when $K \geq 1$ and $n \geq K$. (2) When $n \geq K$, $n \gg 1$ and $K^2 \gg 1$, this ratio will converge to $\frac{K}{2n} + \frac{2}{K}$.

In phase one, after building the initial search space, algorithm SOM will prune the infeasible configurations according to Properties 1-5. Then, algorithm SOM will search the pruned search space for the optimal configuration in phase two.

4.3 Phase Two: Solution Searching

4.3.1 Design of Search Strategy BIS

In phase two of algorithm SOM, a search strategy is employed to search the pruned search space for the optimal configuration. It is obvious that the optimal configuration can be obtained by exhaustive search. However, it is not scalable when the size of the pruned search space is large.

To achieve high scalability, we devise an efficient search strategy, referred to as BIS, based on the analytical models. BIS is a greedy algorithm to find the sub-optimal solution of the search space. In essence, BIS is guided to explore the search space with higher likelihood to be the optimal first. A configuration $C(K_1, n_1)$ is said to be "local optimal when $K_B = K_1$" if $W(K_1, n_1 - 1) \geq W(K_1, n_1)$ and $W(K_1, n_1 + 1) \geq W(K_1, n_1)$. To facilitate the design of BIS, we employ the function *LocalOptimalCheck* to determine whether the input configuration is local optimal. *LocalOptimalCheck*(K_1, n_1) returns `LOCALOPTIMAL` to notify BIS that the input configuration $C(K_1, n_1)$ is the local optimal when $K_B = K_1$. Otherwise, it returns `MINUS` and `PLUS` to show that $W(K_1, n_1 - 1) < W(K_1, n_1)$ and $W(K_1, n_1 + 1) < W(K_1, n_1)$, respectively. The algorithmic form of *LocalOptimalCheck* is as follows.

Function LocalOptimalCheck(K_B, n_B)

1: Calculate(K_B,$n_B - 1$)
2: Calculate(K_B,$n_B + 1$)
3: **if** ($W(K_B, n_B - 1) < W(K_B, n_B)$) **then**
4: return `MINUS`
5: **else if** ($W(K_B, n_B + 1) < W(K_B, n_B)$) **then**
6: return `PLUS`
7: **else** /* $W(K_B, n_B - 1) \geq W(K_B, n_B)$ and $W(K_B, n_B + 1) \geq W(K_B, n_B)$ */
8: return `LOCALOPTIMAL`
9: **end if**

Procedure Calculate(K_B,n_B)

1: Calculate and store $W_B(K_B, n_B)$ and the corresponding broadcast program by employed broadcast program generation algorithm if they had not been calculated
2: Calculate and store $W_O(K - K_B, n - n_B)$ by Equation (1) if it had not been calculated

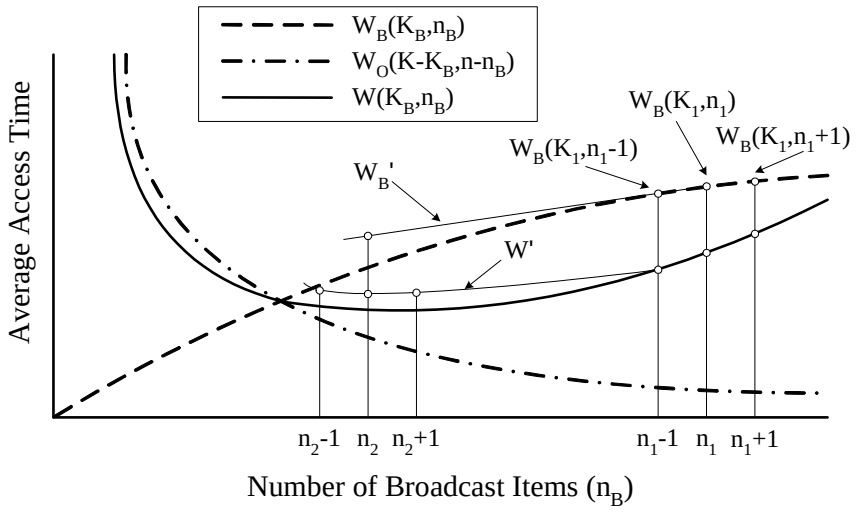


Figure 6: Execution scenario of function *LocalOptimalPrediction*

3: Calculate and store $W(K_B, n_B)$ by Equation (2) if it had not been calculated

Note that each invocation of *LocalOptimalCheck* will cause at least one execution of the broadcast program generation algorithm. That is costly. Therefore, we design function *LocalOptimalPrediction* to predict the position of the local optimal solution to reduce the total execution time by reducing the number of invocations of *LocalOptimalCheck*.

To facilitate the design of function *LocalOptimalPrediction*, we first design a method to calculate the approximations of $W_B(K_B, n_B)$ and $W(K_B, n_B)$. Denote the approximations of $W_B(K_B, n_B)$ and $W(K_B, n_B)$ as $W'_B(K_B, n_B)$ and $W'(K_B, n_B)$, respectively. Figure 6 shows the proposed approximation method which calculates $W'_B(K_B, n_B)$ and $W'(K_B, n_B)$ by extrapolation. As shown in Figure 6, the value of $W'_B(K_1, n_2)$, for each n_2, can be obtained by the extrapolation of $W_B(K_1, n_1)$ and $W_B(K_1, n_1 - 1)$. Then, we have the following equation:

$$\frac{W'_B(K_1, n_2)}{n_2 - n_1} = \frac{W_B(K_1, n_1 + \alpha) - W_B(K_1, n_1)}{\alpha}, \text{ where}$$

$$\alpha = \begin{cases} 1 & : \text{ if } \textit{LocalOptimalCheck}(K_1, n_1) \text{ returns } \texttt{PLUS}, \\ \text{-1} & : \text{ if } \textit{LocalOptimalCheck}(K_1, n_1) \text{ returns } \texttt{MINUS}. \end{cases}$$

By solving the above equation, we have $W'_B(K_1, n_2)$ as:

$$W'_B(K_1, n_2) = \frac{1}{\alpha} \times (n_2 - n_1) \times (W_B(K_1, n_1 + \alpha) - W_B(K_1, n_1)).$$

Since $W_O(K_1, n_2)$ can be obtained by Equation (1), with $W'_B(K_1, n_2)$, $W'(K_1, n_2)$ can be obtained by the following equation:

$$W'(K_1, n_2) = P^n_B(n_2) \times W'_B(K_1, n_2) + (1 - P^n_B(n_2)) \times W_O(K - K_1, n - n_2). \quad (3)$$

LocalOptimalPrediction is employed to predict the position of the local optimal of the configurations with $K_B = K_1$ and $n_{Lower} \leq n_B \leq n_{Upper}$. First, *LocalOptimalPrediction* sets $n_1 = \frac{\lceil n_{Lower} + n_{Upper} \rceil}{2}$ and checks whether $W'(K_1, n_1 - 1) \geq W'(K_1, n_1)$ and $W'(K_1, n_1 + 1) \geq W'(K_1, n_1)$. That is to check whether $W'(K_1, n_1)$ is local optimal. If so, *LocalOptimalPrediction* reports $C(K_1, n_1)$ as the possible configuration of the local optimal solution. Otherwise, if $W'(K_1, n_1 - 1) < W'(K_1, n_1)$, *LocalOptimalPrediction* is invoked recursively by setting $n_{Upper} = n_1 - 1$. Similarly, if $W'(K_1, n_1 + 1) < W'(K_1, n_1)$, *LocalOptimalPrediction* is invoked recursively by setting $n_{Lower} = n_1 + 1$. The algorithmic form of function *LocalOptimalPrediction* is as follows.

Function LocalOptimalPrediction($K_1, n_{Lower}, n_{Upper}$)

1: $n_1 \leftarrow \frac{\lceil n_{Lower} + n_{Upper} \rceil}{2}$
2: Calculate $W'(K_1, n_1)$, $W'(K_1, n_1 - 1)$ and $W'(K_1, n_1 + 1)$ by Equation 3
3: **if** $(W'(K_1, n_1 + 1) < W'(K_1, n_1))$ **then**
4: return LocalOptimalPrediction($K_1, n_1 + 1, n_{Upper}$)
5: **else if** $(W'(K_1, n_1 - 1) < W'(K_1, n_1))$ **then**
6: return LocalOptimalPrediction($K_1, n_{Lower}, n_1 - 1$)
7: **else** /* $W'(K_1, n_1)$ is local optimal */
8: return n_1
9: **end if**

We now design search strategy BIS using *LocalOptimalCheck* and *LocalOptimalPrediction*. After the search space is pruned, BIS checks these unpruned configurations iteratively. In each iteration, BIS picks one value (denoted as K_1) from the possible values of K_B, sets $K_B = K_1$ and considers the configurations with $K_B = K_1$. Suppose that n_{Max} and n_{Min} are the maximum and minimum, respectively, of n_B among all unpruned configurations with $K_B = K_1$. BIS sets $n_1 = \lceil \frac{n_{Max} + n_{Min}}{2} \rceil$ and checks

whether or not the configuration $C(K_1, n_1)$ is the local optimal with $K_B = K_1$ by *LocalOptimalCheck*. If *LocalOptimalCheck* returns `LOCALOPTIMAL`, BIS memorizes configuration $C(K_1, n_1)$ as a candidate of the resultant configuration. Then, BIS steps into next iteration by picking another value of K_1. Otherwise, when *LocalOptimalCheck* returns `PLUS` or `MINUS`, *LocalOptimalPrediction* is invoked to predict the position of the local optimal with $K_B = K_1$. Suppose that *LocalOptimalPrediction* reports that $C(K_1, n_2)$ has the high probability to be the local optimal when $K_B = K_1$. *LocalOptimalCheck* is invoked again to check whether $W(K_1, n_2)$ is the local optimal. In one iteration, BIS repeats the above procedure until the configuration predicted by *LocalOptimalPrediction* is indeed the local optimal (i.e., *LocalOptimalCheck* returns `LOCALOPTIMAL`). After picking all possible values of K_B, BIS stops and returns the best solution among the candidates.

For better understanding of algorithm SOM and search strategy BIS, we design scheme BIS-Generic by employing BIS as the search strategy of algorithm SOM. Without being limited to any broadcast program generation algorithm, scheme BIS-Generic is able to cooperate with any broadcast program generation algorithm seamlessly. The algorithmic form of scheme BIS-Generic is as below, and the procedure of search strategy BIS is described in lines 6-20.

Scheme BIS-Generic
Input: The data items sorted by their access frequencies and the number of communications.
Output: The number of broadcast channels and on-demand channels, the number of data items with broadcast and on-demand channels, and the resultant broadcast program.
Note: Scheme BIS-Generic is not limited to any broadcast program generation algorithm.

1: Construct the search space and prune configurations according to the properties 1-5 /* Phase one */
2: Mark the unavailable configurations (i.e., $K_B > K$ or $K < 0$ or $n_B > n$ or $n_B < 0$) as *calculated* and set $W_B(K_B, n_B)$, $W_O(K - K_B, n - n_B)$ and $W(K_B, n_B)$ to be ∞.
3: **for all** pruned configuration $C(K_B, n_B)$ **do**
4: Set $W_B(K_B, n_B)$, $W_O(K - K_B, n - n_B)$, and $W(K_B, n_B)$ to be ∞ and mark them as *calculated*
5: **end for**
6: **for** ($K_B \leftarrow 0$ to K) **do** /* Phase two */
7: Calculate the corresponding values of n_{Max} and n_{Min}
8: $n_B \leftarrow \frac{\lceil n_{Max} + n_{Min} \rceil}{2}$
9: Calculate(K_B, n_B)
10: **while** (LocalOptimalCheck(K_B, n_B)$\neq$`LOCALOPTIMAL`) **do**
11: **if** (LocalOptimalCheck(K_B, n_B)=`PLUS`) **then**
12: $n_{Min} \leftarrow n_B + 1$
13: $n_B \leftarrow$ LocalOptimalPrediction(K_B, n_{Min}, n_{Max})
14: **else** /* LocalOptimalCheck(K_B, n_B)=`MINUS` */
15: $n_{Max} \leftarrow n_B - 1$
16: $n_B \leftarrow$ LocalOptimalPrediction(K_B, n_{Min}, n_{Max})
17: **end if**
18: **end while**
19: Keep track of the optimal $W_{optimal}(K, n) \leftarrow W(K_B, n_B)$, the corresponding configuration

$C(K_B, n_B)$ and broadcast program
20: **end for**

4.3.2 Employment of the Incremental Property of VF^K

We now design scheme BIS-Incremental, which is able to obtain the local optimal solutions efficiently, by integrating the incremental property of VF^K into scheme BIS-Generic. With the incremental property of VF^K, the execution of VF^K on configuration $C(K_1, n_1)$ will generate K_1 broadcast programs which are the same as the results produced by VF^K for configurations $C(K_B, n_1)$ where K_B =1, 2, 3, $\cdots$, K_1. To take advantage of the incremental property, the search strategy BIS should (1) search K_B in decreasing order and (2) store the results of VF^K obtained by the incremental property for future use. Note that the use of the incremental property of VF^K does not affect the quality of obtained solutions, and VF^K is required to be the broadcast program generation algorithm of scheme BIS-Incremental. The algorithmic form of scheme BIS-Incremental is given below. Since scheme BIS-Incremental is similar to scheme BIS-Generic, only modifications are shown.

Scheme BIS-Incremental
Note: VF^K is required to be the broadcast program generation algorithm.
6′: **for** ($K_B \leftarrow K$ to 0) **do**
Procedure Calculate(K_B, n_B)
1′: Calculate $W_B(K_B, n_B)$ and corresponding broadcast program by VF^K if they had not been calculated. When VF^K is executed, $W_B(\alpha, n_B)$ for all $1 \leq \alpha \leq K_B$ and corresponding broadcast programs are also stored and marked as *calculated*.

4.4 Complexity Analysis

Since the most time-consuming portion of a BIS-based algorithm is the execution of the employed broadcast program generation algorithm, we derive the time complexity of a BIS-based algorithm by focusing on the executions of the employed broadcast program generation algorithm. The time complexity of binary interpolation search in average case is $O(K \log n)$, and therefore, the time complexity of schemes using BIS is "$O(K \log n)\times$ the time complexity of the broadcast program generation algorithm." By employing the incremental property, the *amortized cost* to construct a data broadcast program by VF^K is $\frac{1}{K} \times$ Time Complexity of VF^K. Therefore, the whole time complexity of scheme BIS-Incremental is $O(K \log n) \times \frac{1}{K} \times$ Time Complexity of VF^K $= O(\log n) \times$Time Complexity

Parameter	Value
Number of channels (K)	4
Number of data items (n)	10
Data item request arrival rate (λ)	20/sec
Average waiting time of one on-demand channel ($\frac{1}{\mu}$)	0.1 sec
Average to transfer one data item ($\frac{s}{b}$)	0.1 sec

System parameters

	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_8	R_{10}
$Pr(R_i)$	0.174	0.165	0.147	0.129	0.11	0.092	0.073	0.055	0.037	0.018

Access frequencies

Table 2: An example profile

of VF^K. As shown in [22], with n sorted data items and K broadcast channels given, the time complexity of VF^K is $K \times (O(K \log K) + O(n))$. The time complexity of scheme BIS-Incremental is hence $O(\log n) \times K \times (O(K \log K) + O(n))$. If $n \gg K$, the time complexity of scheme BIS-Incremental is $O(Kn \log n)$. In addition, scheme BIS-Incremental requires a table to store information of each configuration. For K channels and n data items, the size of this table is $(K+1) \times (n+1)$, and hence, the space complexity of scheme BIS-Incremantal is $O(K \times n)$.

4.5 An Illustrative Example

In this subsection, we use a running example to illustrate the steps of scheme BIS-Incremental. Table 2 shows the parameters used in this example. The searching steps are shown in Figure 7 where the number inside a configuration indicates the order of the configuration checked by *LocalOptimalCheck*. The local optimal solution for each value of K_B is marked by thick border.

In phase one, Table 3 is constructed, and $P_B^n(n_B)$ for all $0 \leq n_B \leq 10$ are calculated. Then, configurations are pruned according to Properties 1-5. For each pruned configuration $C(K_B, n_B)$, $W_B(K_B, n_B)$, $W_O(K - K_B, K - n_B)$ and $W(K_B, n_B)$ are initialized to be ∞. Consider the configuration $C(3,3)$. The number of the on-demand channels is $K_O = K - K_B = 4 - 3 = 1$. The data request arrival rate of the on-demand channels is $\lambda_O = \lambda \times P_O^n(3) = 20 \times (1 - P_B^n(3)) = 20 \times (1 - 0.486) = 10.28$. Because $\rho = \frac{\lambda_0}{K_O \mu} = \frac{10.28}{1 \times 10} = 1.028 > 1$, according to Property 5, this configuration is pruned.

In phase two, scheme BIS-Incremental first examines configurations with $K_B = 4$. In this example,

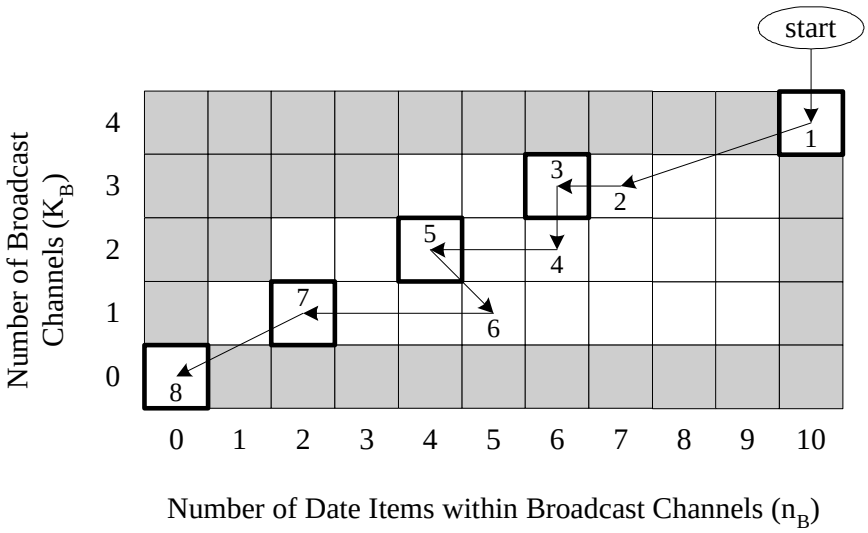


Figure 7: An example search scenario

the only available configuration with $K_B = 4$ is $C(4,10)$. We have $W_O(0,0) = 0$ since it contains no on-demand channel. By executing VF^K, we have $W_B(4,10) = 118$ ms. By Equation (2), we have $W(4,10) = P_B^n(10) \times W_B(4,10) + (1 - P_B^n(10)) \times W_O(0,0) = 118$ ms. Configuration $C(4,10)$ is then checked by *LocalOptimalCheck*. Since configuration $C(4,9)$ is pruned and configuration $C(4,11)$ is unavailable, *LocalOptimalCheck* returns `LOCALOPTIMAL` which means that configuration $C(4,10)$ is local optimal when $K_B = 4$.

Next, configurations with $K_B = 3$ are checked. Since $K_B = 3$, the number of data items on broadcast channels is between 4 and 9. Scheme BIS-Incremental first checks the configuration with $K_B = 3$ and $n_B = \lceil \frac{4+9}{2} \rceil = 7$. $W_O(1,3)$, $W_B(3,7)$ and $W(3,7)$ are then calculated. Due to the incremental property of VF^K, $W_B(2,7)$ and $W_B(1,7)$ are also obtained when VF^K is executed on $K_B = 3$ and $n_B = 7$, and are stored in Table 3b for future use. Note that these two values are not available if other broadcast generation algorithms (e.g., OPT) are employed. Then, $W(3,6)$ and $W(3,8)$ are also calculated in order to check whether $W(3,7)$ is the local optimal when $K_B = 3$. *LocalOptimalCheck*(3,7) returns `MINUS` since $W(3,6) < W(3,7)$. Then, *LocalOptimalPrediction* is invoked and reports that $C(3,6)$ is of high probability to be the local optimal solution. To check whether $W(3,6)$ is indeed the local optimal, VF^K is executed again to obtain $W_B(3,5)$. Finally *LocalOptimalCheck*(3,6) returns `LOCALOPTIMAL` because $W(3,6)$ is less than both $W(3,5)$ and $W(3,7)$.

n_B	0	1	2	3	4	5	6	7	8	9	10
$P_B^n(n_B)$	0	0.174	0.339	0.486	0.615	0.725	0.817	0.89	0.945	0.982	1

(a). Table $P_B^n(n_B)$

	n_B										
K_B	0	1	2	3	4	5	6	7	8	9	10
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	118/0
3	∞	∞	∞	∞	−/−	103/222	110/158	136/128	164/112	−/−	∞
2	∞	∞	50/177	82/136	100/117	127/108	150/103	173/101	199/−	−/−	∞
1	∞	50/122	100/111	150/105	200/102	250/101	300/100	350/−	400/−	−/−	∞
0	0/109	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

(b). Table $W_B(K_B, n_B)/W_O(K-K_B, n-n_B)$

	n_B										
K_B	0	1	2	3	4	5	6	7	8	9	10
4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	118
3	∞	∞	∞	∞	–	136	118	135	161	–	∞
2	∞	∞	134	110	106	122	141	165	–	–	∞
1	∞	109	107	127	162	209	263	–	–	–	∞
0	109	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

(c). Table $W(K_B, n_B)$

Table 3: $W_B(K_B, n_B)$, $W_O(K-K_B, n-n_B)$ and $W(K_B, n_B)$ for the example (time unit: ms)

The same procedure is executed on configurations with K_B =2, 1 and 0, and the results are shown in Table 3. By tracking the optimal configurations in different values of K_B, we can obtain the sub-optimal configuration $C(2,4)$. The configuration $C(2,4)$ means that two channels are operated in broadcast mode and the top four hot data items (i.e., R_1, R_2, R_3 and R_4) are allocated in these two broadcast channels. The remaining channels are operated in on-demand mode and the remaining data items are allocated in the on-demand channels. The broadcast program of these two channels and four data items is obtained by executing VF^K. Finally, the corresponding broadcast program is shown in Figure 8.

5 Performance Evaluation

In order to evaluate the performance improvement achieved by algorithm SOM, we have designed a simulation model of a data dissemination system which is described in Section 5.1. Four schemes are developed based on algorithm SOM to address the problem of dynamic data and channel allocation. Then, four experiments are conducted in the following subsections to examine the impact of different

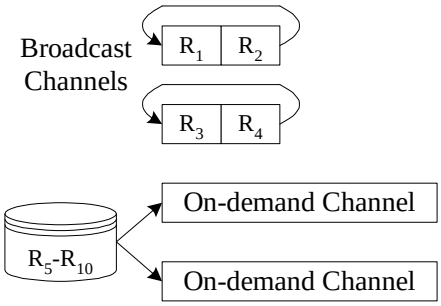


Figure 8: The resultant solution of the running example

Parameters	Values
Channel bandwidth (b)	80000 bps
Data item size (s)	8000 bytes
Data request size (r)	10 bytes
Data request rate for each user	1/sec

Table 4: System parameters used in the simulation

system parameters on the performance of all schemes.

5.1 Simulation Model

Similar to the work in [18][19], we set the system parameters as shown in Table 4. Also, the access frequency of the i-th data item is assumed to be $Pr(R_i) = \frac{(\frac{1}{i})^\theta}{\sum_{j=1}^{n}(\frac{1}{j})^\theta}$ where θ is the parameter of the Zipf distribution. Note that $\theta = 0$ indicates that the access frequencies are uniformly distributed (i.e., $Pr(R_i) = Pr(R_j)$ for all $i,\ j$). In addition, the access frequencies become increasing skewed as θ increases. As pointed out in [8], the value of θ appears to be about 0.8 for traces from a homogeneous environment, and the value of θ appears to be around 0.7 for traces from a diversified user population. In addition, as observed in [21], the value of θ appears to be larger than 1 in busy Web sites. Hence, we set the default value of θ to be 0.9 and conduct an experiment with the value of θ varied to measure the effect of θ. The simulator is coded in C++.

We have implemented four schemes based on algorithm SOM. A scheme denoted by **A-B** means that **A** is the corresponding search strategy and **B** is the corresponding broadcast program generation algorithm. In addition to scheme BIS-Incremental, we implement scheme BIS-VF^K to evaluate the effect of employing the incremental property of VF^K by comparing it with scheme BIS-Incremental.

Scheme	Search Strategy	Broadcast Program Generation Algorithm	Remark
ES-OPT	ES	OPT	
ES-VF^K	ES	VF^K	
BIS-VF^K	BIS	VF^K	
BIS-Incremental	BIS	VF^K	The incremental property of VF^K is employed
FLAT			Scheme FLAT is not an instance of algorithm SOM

Table 5: Schemes in the experiments

Scheme BIS-VF^K is an instance of scheme BIS-Generic by employing VF^K as the broadcast program generation algorithm. To measure the effect of the search strategy, BIS, we also implement scheme ES-VF^K which adopts exhaustive search (abbreviated as ES) and VF^K, respectively, as the search strategy and the broadcast program generation algorithm. For each configuration, since the optimal broadcast programs can be obtained by OPT, the optimal data and channel allocation can be obtained by collecting all optimal broadcast programs of all possible configurations in the search space and taking the optimal one among them. As a result, we implement scheme ES-OPT which employs ES and OPT as the search strategy and the broadcast program generation algorithm, respectively, to obtain the optimal configurations and the corresponding broadcast programs for comparison purposes. Note that all of them are the instances of the proposed algorithm SOM.

In addition to the above SOM-based schemes, scheme FLAT [18], which employs flat broadcast programs (i.e., allocates data items within broadcast channels with equal appearance frequencies), is also implemented in order to evaluate the benefit of using hierarchical broadcast programs. Note that since not being an instance of algorithm SOM, scheme FLAT does not employ any search strategy and broadcast program generation algorithm. A summary of these schemes is given in Table 5.

The following subsections show the average access times and execution times of all schemes on Experiments 1, 2, 3 and 4, respectively, and the parameters of each experiment are listed in Table 6. The ratio of pruned configurations of each scheme is also given to measure the effect these parameters on configuration pruning. Due to the high complexity of OPT, scheme ES-OPT is quite slower than others. Hence, the execution time of scheme ES-OPT is not shown in the following figures. In addition, since scheme ES-VF^K is slower than BIS-based schemes, the execution times of BIS-based schemes are shown in another sub-figures to evaluate the effect of employing the incremental property of VF^K.

	θ	n	K	N
Exp. 1	varied	5000	9	500
Exp. 2	0.9	varied	9	500
Exp. 3	0.9	5000	varied	500
Exp. 4	0.9	5000	9	varied

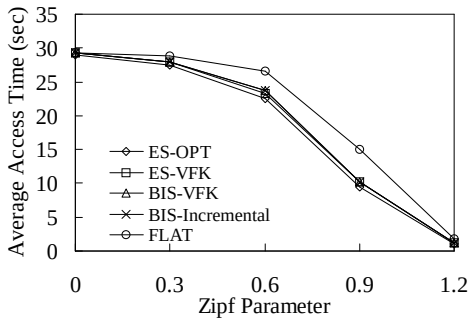
Table 6: Parameters used in the experiments

5.2 Experiment #1: The Effect of the Skewness of Access Frequencies

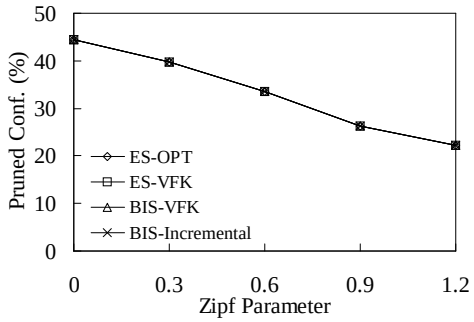
Figure 9 shows the average access times, ratios of pruned configurations and execution times of all schemes with the value of θ varied. The value of θ is set from 0 to 1.2.

As shown in Figure 9a, the average access times of all schemes decrease as the value of θ increases. It can be explained that when the access frequencies are highly skewed, broadcasting hot data items can effectively reduce the load of the on-demand channels, and hence reduce the average access times. We also observe that schemes employing hierarchical broadcast programs (i.e., OPT and VF^K-based schemes) outperform scheme FLAT especially when the access frequencies are highly skewed. In this example, the performance gain of VF^K-based schemes over scheme FLAT increases from 0.5% to 32.14% as the value of θ increases from 0 to 1.2. It fully agrees to the fact that VF^K and OPT outperform FLAT especially when the access frequencies are highly skewed [22]. In addition, the results of VF^K-based schemes are very close to those of scheme ES-OPT (i.e., optimal solutions).

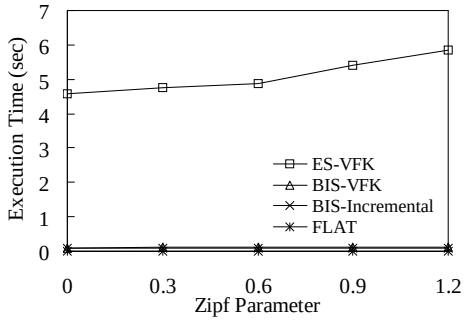
Figure 9b shows the ratio of the pruned configurations with the value of θ varied. Since scheme FLAT does not prune configurations, the pruning effect of scheme FLAT is omitted in this and the following experiments. We observe that the ratio of the pruned configurations decreases from 44.48% to 22.32% as the value of θ increases from 0 to 1.2. Since the number of all configurations and the number of configurations pruned by Properties 1-4 are not affected by the value of θ, this situation results from the pruning effect of Property 5. Note that Property 5 prunes configurations which $\rho \geq 1$. Considering an arbitrary configuration, the condition of $\rho \geq 1$ is when the request rate of the on-demand channels is larger than a threshold (i.e., $\lambda_O \geq K_O\mu$). When θ increases, the access frequencies of cold items decrease. Therefore, on-demand channels can contain more data items without making ρ exceed the threshold. Since the number of configurations pruned by Property 5 decreases as the increase of θ,



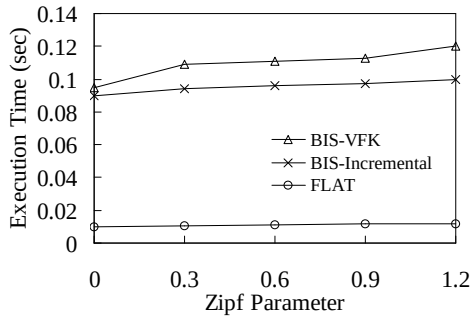
(a) Average Access Time



(b) Ratio of the Pruned Configurations



(c) Execution Time I

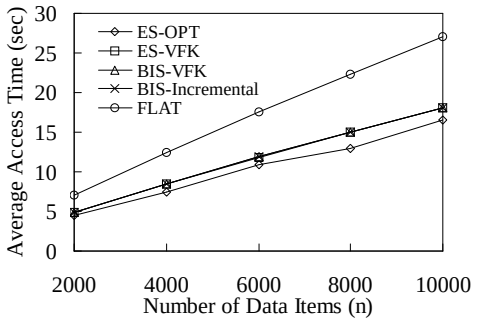


(d) Execution Time II

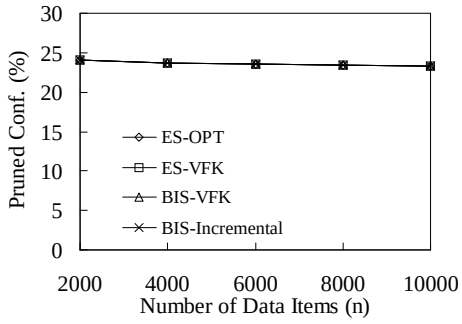
Figure 9: The results with the value of θ varied

the ratio of pruned configurations decreases as the value of θ increases. In addition, since pruning is independent of the employed broadcast program generation algorithms, with the same parameters, the numbers of pruned configurations of all SOM-based schemes are the same.

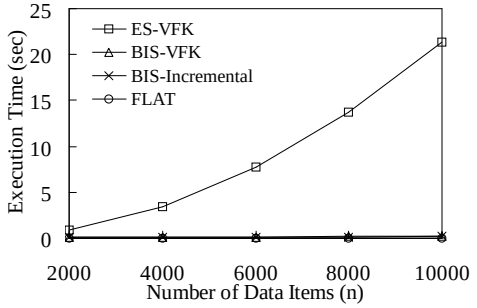
As observed in Figures 9c and 9d, the execution time of scheme ES-VF^K increases as the value of θ increases. It is because that ES examines all unpruned configurations and the effect of configuration pruning decreases as the value of θ increases. On the other hand, since search strategy BIS only checks a subset of unpruned configurations, with the same broadcast program generating algorithm, the execution time of scheme ES-OPT is more sensitive to the change of θ than BIS-based schemes. In this experiment, the execution time reduction of scheme BIS-VF^K over scheme ES-VF^K is around 98%, showing the high efficiency of BIS. In addition, the execution time reduction of scheme BIS-Incremental over scheme BIS-VF^K increases from 5.26% to 20.67% as the value of θ increases from 0 to 1.2. This result shows the advantage of employing the incremental property of VF^K.



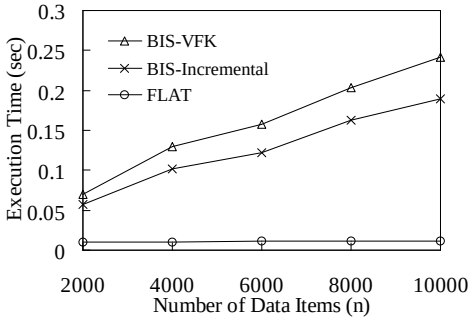
(a) Average Access Time



(b) Ratio of the Pruned Configurations



(c) Execution Time I



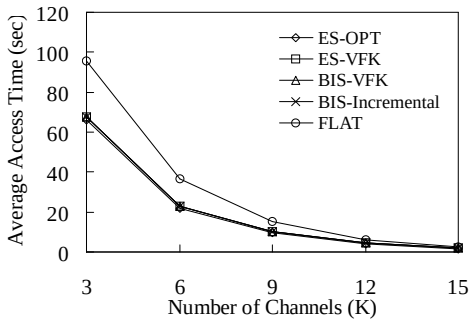
(d) Execution Time II

Figure 10: The results with the number of data items (n) varied

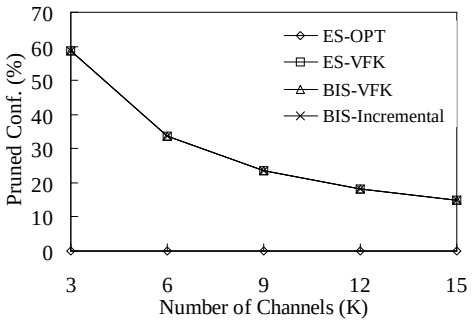
5.3 Experiment #2: The Effect of the Number of Data Items

In experiment 2, we investigate the effect of all schemes with the number of data items (i.e., n) varied. The number of data items is set from 2000 to 10000. As observed in Figure 10a, the average access times of all schemes increase as the number of data items increases. The performance gain of scheme ES-OPT (i.e., optimal solution) over scheme FLAT increases from 35.94% to 39.08% as the number of data items increases, showing the advantage of employing hierarchical broadcast program generation algorithms. In addition, the results of VF^K-based schemes are close to those of scheme ES-OPT, and the performance gain of VF^K-based schemes over scheme FLAT ranges from 30.59% to 33.24%.

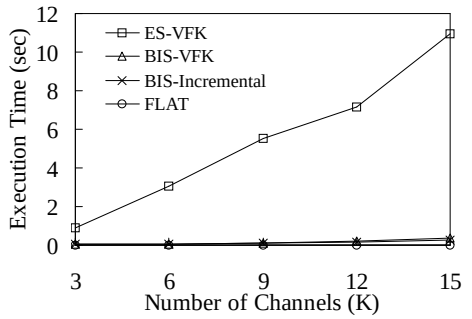
Figure 10b shows that the ratio of pruned configurations slightly decreases from 24.13% to 23.31% as the number of data items increases. This result agrees to the analysis in Lemma 3 that the ratio of pruned configurations is only slightly affected by the value of n since $K \ll n$ in this experiment. In addition, as shown in Figures 10c and 10d, the execution time of each scheme increases with the value of n increases. Although the ratio of pruned configurations only decreases slightly as the number of



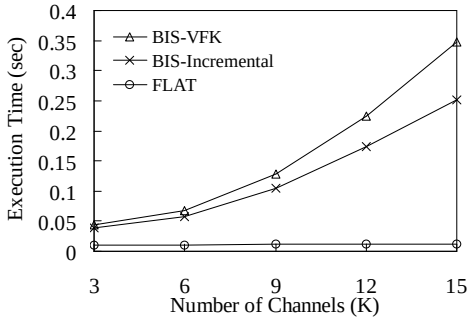
(a) average access time



(b) Ratio of Pruned Configurations



(c) Execution Time I



(d) Execution Time II

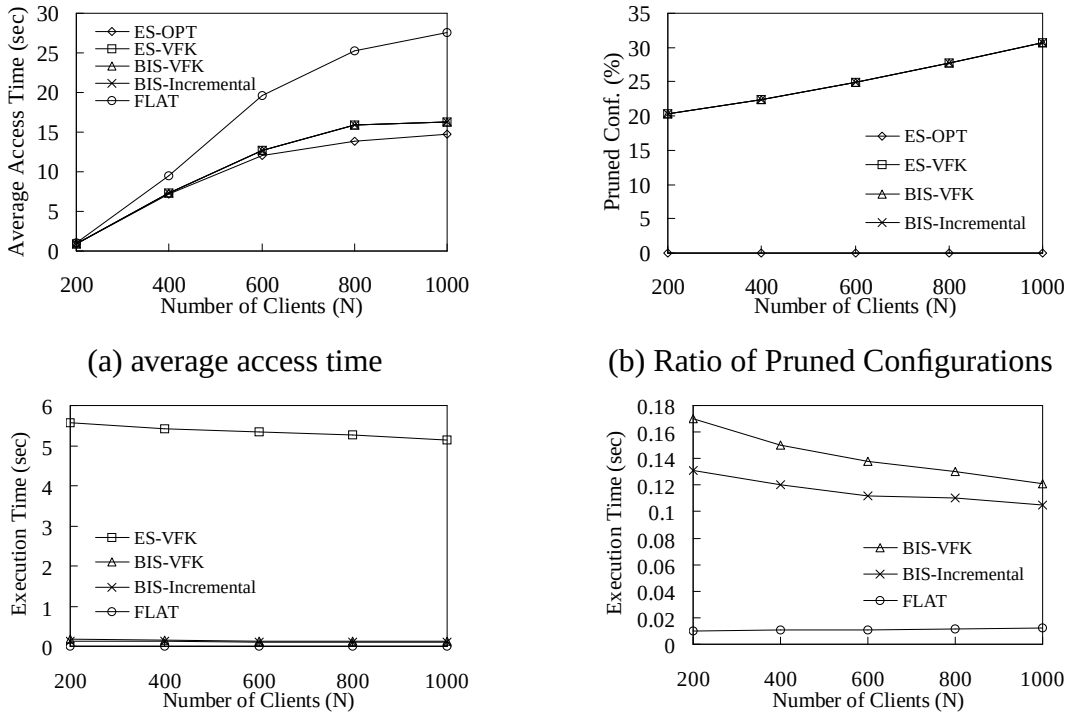
Figure 11: The results with the number of channels (K) varied

data items increases, the increment of the number of unpruned configurations is still in proportion to the increment of the number of data items since the number of all configurations (i.e. $(K+1)\times(n+1)$) increases as the value of n increases. Hence, in execution time, scheme ES-OPT is more sensitive to the number of pruned configurations than BIS-based schemes since scheme ES-OPT scans all unpruned configurations. As a result, BIS-based schemes are more scalable than scheme ES-OPT. As shown in Figures 10c and 10d, as the value of n increases, the execution time reduction of scheme BIS-VF^K over scheme ES-VF^K increases from 93.69% to 99.06%. In addition, the execution time reduction of scheme BIS-Incremental over scheme BIS-VF^K ranges from 18.57% to 22.78%. Since the employment of the incremental property of VF^K does not affect the quality of the results, scheme BIS-Incremental is more scalable than scheme BIS-VF^K on the number of data items.

5.4 Experiment #3: The Effect of the Number of Channels

This experiment evaluates the effect of the number of channels (i.e., K) which is set from 3 to 15. As shown in Figure 11a, the average access times of all schemes decrease as the number of channels increases. This result agrees to the intuition that the increase of bandwidth will decrease the average access time. However, the improvement on the average access time decreases as the number of channels increases. As a result, the determination of the number of channels should consider the balance between performance improvement and the number of channels used. We also observe that the performance gain of scheme ES-OPT over scheme FLAT ranges from 25.02% to 38.26% as the number of channels increases. In addition, the performance gain of VF^K-based schemes over scheme FLAT ranges from 19.03% to 36.87%. These results show that the schemes employing hierarchical broadcast programs are able to utilize network bandwidth better than that employing flat broadcast programs.

As shown in Figure 11b, the ratio of pruned configurations decreases from 58.63% to 14.83% as the number of channels increases. With the analysis in Lemma 3, the influence of the ratio of pruned configurations is dominated by K rather than n since $K \ll n$ in this experiment. As a result, the influence of the change of K is more significant than that of the change of n. Figures 11c and 11d show that the execution times of all schemes increase as the number of channels increases. It can be explained as follows. The execution times of all schemes are proportional of the number of unpruned configurations, which increases as the value of K increases since the number of all configurations is $(K+1)\times(n+1)$ and the ratio of pruned configurations decreases as the value of K increases. Since the execution time of scheme ES-OPT is more sensitive to the number of unpruned configurations than that of BIS-based schemes, BIS-based schemes are more scalable when the value of K becomes large. As shown in Figures 11c and 11d, the execution time reduction of scheme BIS-VF^K over scheme ES-VF^K ranges from 96.04% to 98.01% as the value of K increases. In addition, the execution time reduction of scheme BIS-Incremental over scheme BIS-VF^K increases from 11.36% to 27.87%. This result shows the high scalability of scheme BIS-Incremental.



(a) average access time

(b) Ratio of Pruned Configurations

(c) Execution Time I

(d) Execution Time II

Figure 12: The results with the number of users (N) varied

5.5 Experiment #4: The Effect of the Number of Users

This experiment measures the effect of the number of users. Figure 12a shows the average access times of all schemes with the number of users varied. The number of users is set from 200 to 1000. As observed in Figure 12a, the average access time reduction of scheme ES-OPT over scheme FLAT increases from 11.8% to 40.66% as the number of users increases. It is because that the data request rate is proportional of the number of users. In addition, the increment of average access time decreases as the number of users increases. It can be explained as follows. When the number of users is small, most channels are allocated in on-demand mode and most data items are allocated in the on-demand channels. When the number of users becomes large, to reduce the increment of average access time, some channels are re-allocated to broadcast mode and some data items are re-allocated to the broadcast channels. Hence, the system becomes less sensitive to the number of users when the number of users increases. This result shows the advantage of the combined use of broadcast and on-demand channels. In this experiment, the performance gain of scheme ES-OPT over scheme FLAT increases from 1.10%

to 46.20% as the number of users increases, and the performance gain of VF^K-based schemes over scheme FLAT ranges from 0.81% to 40.66%.

Figure 12b shows that the ratio of pruned configurations increases from 20.36% to 30.71% as the value of N increases. Similar to the situation when θ varies, the increase of the value of N causes more configurations to be pruned by Property 5. Hence, the ratio of pruned configurations increases as the value of N increases. It shows that the pruning properties are scalable when the number of users is high.

Figures 12c and 12d show the execution time of each scheme with the value of N varied. Resulting from the effect showing in Figure 12b, the execution times of all schemes decrease as the value of N increases. In addition, since BIS-based schemes are less sensitive to the number of unpruned configurations than scheme ES-OPT, the increment of the execution times of BIS-based schemes is smoother than that of scheme ES-OPT. In this experiment, the execution time reduction of scheme BIS-VF^K over scheme ES-VF^K is around 97%, and the execution time reduction of scheme BIS-Incremental over scheme BIS-VF^K decreases from 22.94% to 13.22%.

5.6 Summary

In this section, we evaluate the performance of several instances of algorithm SOM. From above experiments, we observe that the average access time of all schemes employing hierarchical broadcast generation programs (i.e., OPT and VF^K-based schemes) is better than that of scheme FLAT which employs flat broadcast programs. This result shows the advantage of using hierarchical broadcast program generation algorithms. The solutions obtained by VF^K-based schemes are close to scheme ES-OPT due to the fact that the results of VF^K are close to those of OPT.

We also observe that the execution time of BIS-based schemes is much faster than that of scheme ES-OPT when the same broadcast program generation algorithm is employed. It is because that BIS only searches the configurations with high probability to be the optimal one instead of all configurations in the search space. Due to the combination of the merits of BIS and VF^K, scheme BIS-VF^K is able to obtain nearly-optimal solutions efficiently. In addition, by employing the incremental property of VF^K, scheme BIS-Incremental is able to obtain the same solutions as what scheme BIS-VF^K obtains and is more efficient and scalable than scheme BIS-VF^K.

6 Conclusions

In this paper, we explored the problem of dynamic data and channel allocation with the number of communication channels and the number of data items given. We first derived the analytical models of the average access time on broadcast and on-demand channels. Then, we transformed this problem into a guided search problem. In light of the theoretical properties derived, we devised algorithm SOM to obtain the optimal allocation of data and channels. According to the analytical mode, we devised scheme BIS-Incremental based on SOM which is able to obtain solutions of high quality efficiently by employing binary interpolation search and the incremental property of VF^K. Sensitivity study on several parameters, including the number of data items and the number of communication channels, was conducted. Our simulation results showed that the solutions of scheme BIS-Incremental are of very high quality and are in fact very close to the optimal ones. In addition, the experimental results also showed that scheme BIS-Incremental is of very good scalability which is particularly important for its practical use in a mobile computing environment.

References

[1] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik. Broadcast Disks: Data Management for Asymmetric Communication Environments. In *Proceedings of the ACM SIGMOD Conference*, pages 198–210, March 1995.

[2] S. Acharya, M. Franklin, and S. Zdonik. Balancing Push and Pull for Data Broadcast. In *Proceedings of the ACM SIGMOD Conference*, pages 183–194, May 1997.

[3] S. Acharya and S. Muthukrishnan. Scheduling on-demand Broadcasts: New Metrics and Algorithms. In *Proceedings of the 4th ACM/IEEE International Conference on Mobile Computing and Networking*, pages 43–94, October 1998.

[4] M. Agrawal, A. Manjhi, N. Bansal, and S. Seshan. Improving Web Performance in Broadcast-Unicast Networks. In *Proceedings of the IEEE INFOCOM Conference*, March-April 2003.

[5] D. Aksoy and M. J. Franklin. Scheduling for Large-Scale On-Demand Data Broadcasting. In *Proceedings of IEEE INFOCOM Conference*, pages 651–659, March 1998.

[6] D. Aksoy, M. J. Franklin, and S. Zdonik. Data Staging for On-Demand Broadcst. In *Proceedings of the 27th International Conference on Very Large Data Bases*, pages 571–580, September 2001.

[7] A. Bar-Noy, B. Patt-Shamir, and I. Ziper. Broadcast Disks with Polynomial Cost Functions. *ACM/Kluwer Wireless Networks*, 10(2), March 2004.

[8] L. Breslau, P. Cao, G. Phillips, and S. Shenker. Web Caching and Zipf-like Distributions: Evidence and Implications. In *Proceedings of the IEEE INFOCOM Conference*, March 1999.

[9] M.-S. Chen, K.-L. Wu, and P. S. Yu. Indexed Sequential Data Broadcasting in a Wireless Computing Environment. In *Proceedings of the 17th IEEE International Conference on Distributed Computing Systems*, pages 124–131, May 1997.

[10] A. Datta, D. E. VanderMeer, A. Celik, and V. Kumar. Broadcast Protocols to Support Efficient Retrieval from Databases by Mobile Users. *ACM Transactions on Database Systems*, 24(1):1–79, March 1999.

[11] D. Gross and C. M. Harris. *Fundamentals of Queueing Theory*. John Wiley & Sons,Inc., 3rd edition, 1998.

[12] C.-H. Hsu, G. Lee, and A. L. P. Chen. A Near Optimal Algorithm for Generating Broadcast Programs on Multiple Channels. In *Proceedings of the 10th ACM International Conference on Information and Knowledge Management*, November 2001.

[13] C.-L. Hu and M.-S. Chen. Dynamic Data Broadcasting with Traffic Awareness. In *Proceedings of the 22th IEEE International Conference on Distributed Computing and Systems*, July 2002.

[14] Q. L. Hu, W.-C. Lee, and D. L. Lee. Indexing Techniques for Wireless Data Broadcast under Data Clustering and Scheduling. In *Proceedings of the 8th ACM International Conference on Information and Knowledge Management*, pages 351–718, November 1999.

[15] T. Imielinski, S. Viswanathan, and B. R. Badrinath. Data on Air: Organization and Access. *IEEE Transactions on Knowledge and Data Engineering*, 9(9):353–372, June 1997.

[16] J. Juran, A. R. Hurson, N. Vijaykrishnan, and S. Kim. Data Organization and Retrieval on Parallel Air Channels: Performance and Energy Issues. *ACM/Kluwer Wireless Networks*, 10(2), March 2004.

[17] S. Lee, D. P. Carney, and S. Zdonik. Index Hint for On-demand Broadcasting. In *Proceedings of the 19th IEEE International Conference on Data Engineering*, March 2003.

[18] W.-C. Lee, Q. L. Hu, and D. L. Lee. A Study on Channel Allocation for Data Dissemination in Mobile Computing Environments. *ACM/Kluwer Mobile Networks and Applications*, 4(5):117–129, May 1999.

[19] C.-W. Lin, H. Hu, and D. L. Lee. Adaptive Realtime Bandwidth Allocation for Wireless Data Delivery. *ACM/Kluwer Wireless Networks*, 10:103–120, 2004.

[20] S.-C. Lo and A. L. P. Chen. Optimal Index and Data Allocation in Multiple Broadcast Channels. In *Proceedings of the 16th International Conference on Data Engineering*, pages 293–702, March 2000.

[21] V. Padmanabhan and L. Qiu. The Content and Access Dynamics of a Busy Web Site: Findings and Implications. In *Proceedings of the IEEE SIGCOMM Conference*, pages 293–304, August-September 2000.

[22] W.-C. Peng and M.-S. Chen. Efficient Channel Allocation Tree Generation for Data Broadcasting in a Mobile Computing Environment. *ACM/Kluwer Wireless Networks*, 9(2):117–129, 2003.

[23] K. Prabhakara, K. A. Hua, and J. H. Oh. Multi-Level Multi-Channel Air Cache Designs for Broadcasting in a Mobile Environment. In *Proceedings of the 16th International Conference on Data Engineering*, pages 167–186, February-March 2000.

[24] M. Satyanarayanan. Pervasive Computing: Vision and Challenges. *IEEE Personal Communications*, 8(4):10–17, August 2001.

[25] N. Shivakumar and S. Venkatasubramanian. Efficient Indexing for Broadcast Based Wireless Systems. *ACM/Baltzer Mobile Networks and Applications*, 4(6):433–446, January 1996.

[26] K. Stathatos, N. Roussopoulos, and J. S. Baras. Adaptive Data Broadcast in Hybrid Networks. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, pages 326–335, 1997.

[27] C.-J. Su and L. Tassiulas. Joint Broadcast Scheduling and User's Cache Management for Efficient Information Delivery. In *Proceedings of the 4th ACM/IEEE International Conference on Mobile Computing and Networking*, pages 33–42, October 1998.

[28] D. A. Tran, K. Hua, and K. Prabhakaran. On The Efficient Use of Multiple Physical Channel Air Cache. In *Proceedings of the IEEE Wireless Communications and Networking Conference*, 17-21 2002.

[29] WAP Forum. http://www.wapforum.org/.

[30] J. Xu, W.-C. Lee, and X. Tang. Exponential Index: A Parameterized Distributed Indexing Scheme for Data on Air. In *Proceedings of the 2nd ACM/USENIX International Conference on Mobile Systems*, June 2004.

[31] J. L. Xu, Q. L. Hu, W.-C. Lee, and D. L. Lee. An Optimal Cache Replacement Policy for Wireless Data Dissemination under Cache Consistency. In *Proceedings of the 30th International Conference on Parallel Processing*, September 2001.

[32] J. L. Xu, D. L. Lee, and B. Li. On Bandwidth Allocation for Data Dissemination in Celluar Mobile Networks. *ACM/Kluwer Wireless Networks*, 9(2):103–116, March 2003.

[33] J. L. Xu, B. Zheng, W.-C. Lee, and D. K. Lee. Energy Efficient Index for Querying Location-Dependent Data in Mobile Broadcast Environments. In *Proceedings of the 19th International Conference on Data Engineering*, March 2003.

[34] W. G. Yee, S. B. Navathe, E. Omiecinski, and C. Jermaine. Bridging the Gap between Response Time and Energy-Efficiency in Broadcast Schedule Design. In *Proceedings of the International Conference on Extending Data Base Technology*, pages 572–589, 2002.

[35] W. G. Yee, S. B. Navathe, E. Omiecinski, and C. Jermaine. Efficient Data Allocation Over Multiple Channels at Broadcast Servers. *IEEE Transactions on Computers*, 51(10):1231–1236, October 2002.

[36] J. X. Yu, T. Sakata, and K. L. Tan. Statistical Estimation of Access Frequencies in Data Broadcasting Environments. *ACM/Kluwer Wireless Networks*, 6(2):89–98, March 2000.

Appendix: All Proof

Proof of Property 1: Consider an arbitrary configuration C which $1 \leq K_B \leq K-1$ and $n_B < K_B$. Since $n_B < K_B$, at least one broadcast channel does not contain any data item. Then, we can get another configuration C$'$ by reassigning the broadcast channel(s) without any data item as on-demand channel(s). P_B^n is equal to $P_B'^n$ since no data item is reassigned. Since these reassigned broadcast channels contain no data item, the average access times in broadcast channels of C and C$'$ are equal (i.e., $W_B = W_B'$). Since C$'$ has more on-demand channels than C, W_O' is smaller than W_O. By Equation (2), we have $W' < W$, and as a result, C is not the optimal since C$'$ is better than C. **Q.E.D.**

Proof of Lemma 1: Consider the procedure of VF^K mentioned above. The initial partitions of all configurations with the same parameters except K_B are the same (i.e., placing all data items in one partition). Then, the selected partitions to be cut and the best cut points for these configurations are the same. Hence, the results after the first cuts of all configurations with the same parameters except K_B equal to the result of VF^K when $K_B = 2$. With the same reasoning, the results of the n-th cuts of all configurations with the same parameters except K_B equal to the result of VF^K when $K_B = n+1$. This property follows. **Q.E.D.**

Proof of Lemma 2: When $K = 1$, the size of search space is $(K+1) \times (n+1) = (1+1) \times (n+1) = 2n+2$. The feasible configurations are $C(1,n)$ and $C(0,0)$. Then, the number of configurations pruned by Properties 1-4 is $2n+2-2 = 2n$.

Considering the cases that $K > 1$, when $K_B = 0$, Properties 2 and 3 are able to prune n configurations. For each K_B, $1 \leq K_B \leq K-1$, Properties 1 and 2 are able to prune K_B and one configurations respectively. When $K_B = K$, Properties 1 and 4 are able to prune n configurations. Then, the total number of configurations pruned by Properties 1 to 4 is

$$\begin{aligned} & \text{Number of configurations pruned by Properties 1-4} \\ = \; & n + \sum_{i=1}^{K-1} (i+1) + n \\ = \; & 2n + \frac{(K-1)(K+2)}{2}. \end{aligned}$$

Consequently, we can conclude that the total number of configurations pruned by Properties 1-4 is $2n + \frac{(K-1)(K+2)}{2}$. **Q.E.D.**

Proof of Lemma 3: Initially, the total number of configurations in the search space is $(n+1)(K+1)$. When $K \geq 1$ and $n \geq K$, according to Lemma 2, the number of configurations pruned by Properties 1-5 is at least $2n + \frac{(K-1)(K+2)}{2}$. Then, the lower bound of the ratio of the pruned configurations can be formulated as follows:

$$\begin{aligned} & \text{The lower bound of the ratio of the pruned configurations} \\ \geq\ & \frac{2n + \frac{(K-1)(K+2)}{2}}{(n+1)(K+1)} \\ =\ & \frac{4n + K^2 + K - 2}{2(n+1)(K+1)} \end{aligned}$$

When $n \gg 1$ and $K^2 \gg 1$,

$$\begin{aligned} & \text{The lower bound of the ratio of the pruned configurations} \\ \geq\ & \frac{4n + K^2 + K - 2}{2(n+1)(K+1)} \\ \approx\ & \frac{4n + K^2 + K + 4}{2(n+1)(K+1)} \\ =\ & \frac{1}{2} \times \left(\frac{K}{n+1} + \frac{4}{K+1} \right) \end{aligned}$$

Note that $\frac{1}{K+1} \approx \frac{1}{K}$ when $K^2 \gg 1$. The approximated lower bound of the ratio of the pruned configurations when $n \gg 1$ and $K^2 \gg 1$ is

$$\begin{aligned} & \text{The ratio of pruned configurations} \\ \geq\ & \frac{1}{2} \times \left(\frac{K}{n+1} + \frac{4}{K+1} \right) \qquad \text{since } n \gg 1 \text{ and } K^2 \gg 1 \\ \approx\ & \frac{1}{2} \times \left(\frac{K}{n} + \frac{4}{K} \right) \\ =\ & \frac{K}{2n} + \frac{2}{K}, \end{aligned}$$

proving Lemma 3. **Q.E.D.**