2012 Intelligent Vehicles Symposium
Alcalá de Henares, Spain, June 3-7, 2012

Distributed Skip Air Index for Smart Broadcasting in Intelligent Transportation Systems

Leandros A. Maglaras Dimitrios Katsaros
Department of Computer & Communication Engineering, University of Thessaly, Volos, Greece
http://www.inf.uth.gr/∼dkatsar

Abstract— **Wireless data broadcast received a lot of attention from industries and academia in recent years. In any form of a push-based broadcast, access latency and tuning time are vital issues, and in order to address the tradeoff among these competing goals, the broadcasting of indices along with the data is the most viable solution. Currently, two broad indexing families exist: those that exploit some form of a tree structure, and those that are based on some 'distributed access on the air' mechanism. The latter family is the most popular and viable, because it allows for following 'air-pointers' without the need to first find a tree root. The champion method of the distributed air index is the** *Exponential index* **which however is not appropriate when the access pattern is skewed, i.e., some data items are more popular than the others. To address this shortcoming, we design a** *Distributed Skip Air Index* **($DiSAIn$), which exploits access statistics in order to improve average tuning time, while it preserves the access latency equal to that of the original Exponential index. To attest the superiority of the proposed indexing method, we perform a detailed simulation evaluation of the two competing methods.**

I. Introduction

A wealth of wireless networks is nowadays present such as cellular, WiMAX, vehicle-to-infrastructure, and so on. In all these types of communication systems, broadcasting is the fundamental operation that exploits the shared medium, i.e., the wireless channel, to transmit information to clients. The communication protocols in such architectures can be implemented either as pure pull-based, pure push-based [20] or on-demand broadcasting [1], [26]. In pure pull-based broadcast, the clients request information via an uplink channel, and subsequently the server allocates a channel for the requesting client, and transmits the information. In pure push-based broadcast, the server sends information over a common broadcast channel to all listening 'consumers' (clients). In on-demand broadcast, the clients pose requests to the server, and the server broadcasts the responses via a shared broadcast channel – thus a single response can satisfy multiple requests. Apparently, (pure-push and on-demand) broadcasting is a preferred choice for modern wireless networks, since it overcomes the scalability issues associated with the large number of consumers and the large volume of transmitted data.

Consider for instance a data dissemination [7] application in an Infrastructure-to-Vehicle (I2V) case. In this push-based broadcast system, the RSUs broadcast information concerning issues relevant to the moving vehicles, e.g., traffic congestion reports, updated routing instructions for the vehicles of a fleet, and so on. Each RSU constructs a broadcast program with the needed info and broadcasts it periodically. All vehicles can tune in to the broadcast channel in order to retrieve the information (data items) without sending explicit requests for them, thus avoid 'choking' the uplink channel. Figure 1 illustrates such a scenario. A vehicle wants to retrieve data item *B* from a broadcast channel. The importance of knowing when item *B* will be broadcasted is crucial, because the vehicle can decide to accelerate to get it from the next RSU, or slow down to get it from the current RSU. The presence of *air indexes* would give an answer about the time of broadcast. Regulation of the vehicle's velocity and knowledge of the distance between successive RSUs could provide a suggestion to the driver about its driving policy.

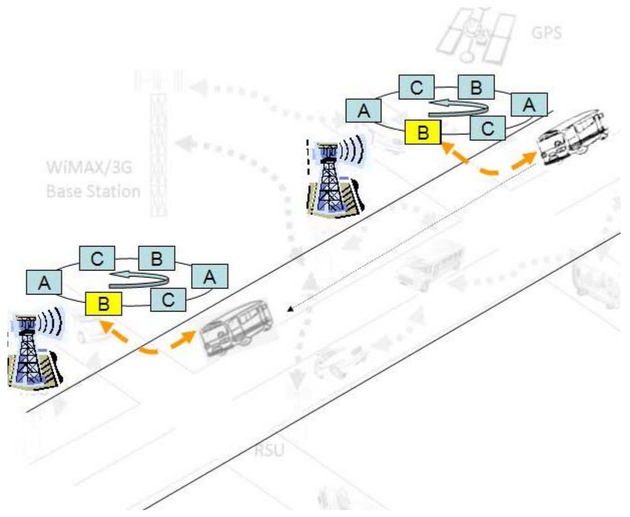


Fig. 1. I2V type of communication.

Such information is important in an overall CO_2 emission control policy. The more accurate the information about the place of the desired data in the overall broadcast cycle is, the less alteration (acceleration or slow down) in the velocity of the vehicle is required in order to be in range of an RSU for the whole time period the desired data is broadcasted. Optimizing driving behavior patterns, such as idling, speeding, fast stops and hard braking, have a direct impact on fuel economy

Research supported by the project "REDUCTION: Reducing Environmental Footprint based on Multi-Modal Fleet management System for Eco-Routing and Driver Behaviour Adaptation", funded by the EU.ICT program, Challenge ICT-2011.7.

and CO_2 emissions which is a great challenge[1] that the transportation section faces today. Besides, such information is also useful for hybrid geocast routing protocols [2], for use by dead drops [5] and so on.

In most situations the majority of the vehicles that move in a specified area tend to retrieve some particular data instead of showing equal preference to all data; therefore the skewness in data access is very high [15]. An index that help the listener find faster this information would definitely improve the driving behavior patterns.

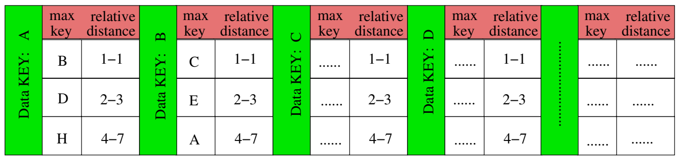
To overcome that problem of 'blindly' searching among all transmitted data, the servers usually interleave along with the data, and some *index* packets, to help the consumers 'move' into the broadcasted information. The index packets contain the necessary information to guide the user through the transmitted data, until s/he reaches the information. These index packets play the role of the indices that we encounter into the traditional disk-based database systems. The difference between the two is that the broadcast channel is a *one-dimensional medium*, and the interleaving of index packets increases the *access time*, in an attempt to decrease the *tuning time*. These performance measures are typically used to measure the efficiency of a wireless data broadcast system; access time is the time elapsed from the moment a request is issued by a client to the moment the requested data is returned, and tuning time is the duration of time a client stays tuned in to collect requested data items. All indexing schemes attempt to achieve the best tradeoff between tuning time and access time.

A. Motivation

A lot of indexing schemes have been proposed in the literature (cf. Section V for a detailed survey); they can be categorized as *hierarchical tree-based* and *distributed* solutions. Examples of the first family include an adaptation of the idea of B$^+$-tree indexing in wireless environments [11], and application of signature trees as indexing methods [14]. B$^+$-trees and hashing are appropriate for uniform access patterns. To deal with skew access patterns, other tree-based methods include [13], [19]. Adaptation of such indexing schemes (e.g., B$^+$-tree) to work in multiple broadcast channels are described in [8], [25].

Distributed solutions such as the adaptation of the traditional hash-based indexing technique in wireless environments was earlier described in [10], and later was generalized in [21], [22], which can be seen as a distributed air-based implementation of a skip list. Hybrids between the two families are described in [18], [24]. In summary, any method that is (completely or partially) based on a broadcasted tree is inferior, compared to the distributed solutions, because the users can not start their search immediately after they tune into the broadcast channel.

In [21], Xu et al. proposed the exponential index, $ExpIn$. In $ExpIn$, each bucket contains a data part and an index table. The index table consists of a number of rows which varies according to the database size. Each entry indexes a segment of buckets with sizes that grow exponentially (i.e., $2^0, 2^1, 2^2, \cdots$). The first entry contains the next bucket ($2^0 = 1$) and for each $i > 1$ the i_{th} entry points to the segment of buckets that are 2^{i-1} to 2^i away (see Figure 2). Instead of the base 2, the algorithms can use any integer number, but this achieves an acceptable tradeoff between access time and tuning time. For more details, the interested reader can resort to [22].



Data KEY: A	max key	relative distance	Data KEY: B	max key	relative distance	Data KEY: C	max key	relative distance	Data KEY: D	max key	relative distance	....	max key	relative distance
	B	1–1		C	1–1		......	1–1		......	1–1		......	......
	D	2–3		E	2–3		......	2–3		......	2–3		......	......
	H	4–7		A	4–7		......	4–7		......	4–7		......	......

Fig. 2. Exponential index structure for a sample database consisting of the elements 'A'–'H'.

In [21], [22] it was shown that the *exponential index* ($ExpIn$) is superior to the hashing schemes of [10] (and to the tree-based schemes), and currently this indexing scheme is the state-of-the-art for broadcast indexing. However, the exponential index is able to achieve good tuning time only when the broadcasted data have equal probability of access. In real life applications though, some items are more frequently accessed by clients than some others (the less popular ones); i.e. a *skewed access pattern* [4] is prevalent in the real life. This feature makes *$ExpIn$'s performance degrades proportionally to the skewness of data*, and it comprises a motivation of the present work.

In this article, we develop an indexing scheme for a single broadcast channel, suitable for skewed access patterns that is distributed in its nature, i.e., no matters when a user tunes into the broadcast channel, s/he can immediately start the searching for the desired item. Specifically, the article makes the following contributions:

- It develops a broadcast index, namely the *Distributed Skip Air Index* (*$DiSAIn$*), which exploits the different access probabilities of data items, in order to improve the mean tuning time, while retaining the same access time as that of the exponential indexing scheme.
- It evaluates experimentally the performance of the proposed broadcast index, against the Exponential Index, for several database sizes and access probabilities distributions.

The rest of this article is organized as follows: Section II describes the network model, i.e., any assumptions made in the present work; Section III presents the Distributed Skip Air Index; Section IV presents the simulation environment, the experiments and obtained results. Finally, Section V briefly survey the most relevant work, and Section VI concludes the article.

II. Network model

We will abstract our application area with the generic model which is depicted in Figure 3. We will use the term 'client'

[1]http://www.reduction-project.eu/

to any device, e.g., vehicle, smart phone, that is interested to get information from the broadcast channel.

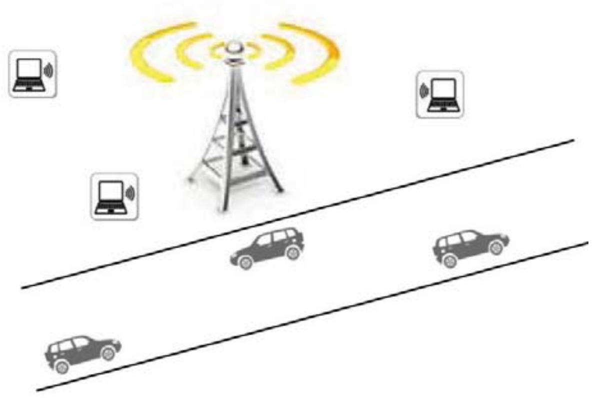


Fig. 3. Generic architecture of broadcast-based wireless networks.

We will consider a single channel wireless broadcast environment with the following generic properties:

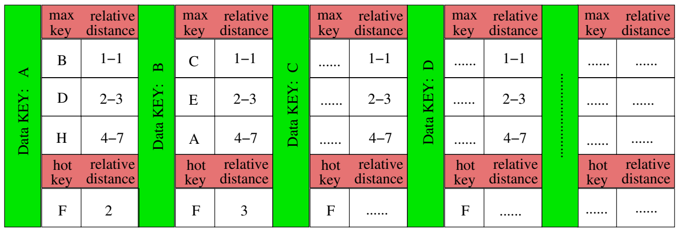
- The broadcast schedule is push-based.
- Data access patterns are skewed.
- All data items are of equal size.
- Time on the broadcast channel is divided into slots where each slot can accommodate one data item.
- The server broadcasts data items in the range of 1 to database size.
- The server retains access statistics.
- Clients require only one item at one time, i.e. a single-item query.
- The client after acquiring the desired data, remains idle for an exponentially distributed time period before next data request.
- Initiation phase takes place every N broadcast cycles in order to stay updated to variations to data access probability distribution.

III. The Distributed Skip Air Index ($DiSAIn$)

The objective of the $DiSAIn$ index is to address the problem of skewed access probabilities observed in real life applications and the appropriate data dissemination in VANETs. To achieve this, it adds more 'pointers' to the original $ExpIn$ index, but it retains some of the properties of the $ExpIn$ index: each bucket of the broadcast cycle contains a data bucket and an index table. The index table consists of i entries. Each entry indexes a range of buckets that are 2^{i-1} to $2^i - 1$ buckets away and holds the maxkey value of these buckets. But for the last bucket, the $DiSAIn$ maintains another index, the skewed index SI, that points to the most popular element MP of the segment. The construction of the index table takes place in an initiation phase where the distance of MP from the *maxkey* value of the last segment is calculated for each data element. In case where all data elements of the last segment have equal access probabilities, then the $DiSAIn$ indexes one of these elements at random.

Figure 4 illustrates a sample $DiSAIn$ index; it is supposed that the elements 'A' to 'H' are to be indexed, and that the element 'F' is the most popular one. In general, one fourth of the total number of pointers point to this 'hot' element. If the client tunes into the broadcast channel just before item 'A', then it retrieves the index table that corresponds to the bucket of 'A'. In case where the client issues a query for item 'F', a pointer points to it directly, minimizing the tuning time while keeping the access latency constant.

The effectiveness and efficiency of the $DiSAIn$ index is not only based on the direct indexing of the most popular item(s), which creates huge earnings, since in real environments most of the clients retrieve hot items. Even in cases where clients are not interested in the popular ones, this additional pointer shortens the tuning time, since it provides evidence about the relative position of other items. For instance, looking at the index that corresponds to element 'A', the client can deduce that between elements 'F' (five places away) and element 'H' (seven places away) there is one element in-between them, therefore this is the element 'G'.



Data KEY: A
max key | relative distance
B | 1–1
D | 2–3
H | 4–7
hot key | relative distance
F | 2
Data KEY: B
max key | relative distance
C | 1–1
E | 2–3
A | 4–7
hot key | relative distance
F | 3
Data KEY: C
max key | relative distance
...... | 1–1
...... | 2–3
...... | 4–7
hot key | relative distance
F | ......
Data KEY: D
max key | relative distance
...... | 1–1
...... | 2–3
...... | 4–7
hot key | relative distance
F | ......
....
max key | relative distance
...... | ......
...... | ......
...... | ......
hot key | relative distance
...... | ......

Fig. 4. A small example of the $DiSAIn$ indexing strategy.

In the previous discussion, we described the $DiSAIn$ index as a plain addition of one pointer to the $ExpIn$ index. Actually, the real flesh of the index, is the addition of more pointers towards the rest of the popular items. Clearly, there is a tradeoff and an optimal value of added pointers. We refer to this version of the index as the *Enhanced* $DiSAIn$. In this paper, for the interest of space, we do not examine further this generalized version, but stick to the most simple one, since even this naive approach is able to reap significant performance gains, as this is presented in the next section. The theoretical and experimental analysis of the generalized version will be done elsewhere.

IV. Evaluation

For the evaluation of the proposed indexing scheme against the state-of-the-art broadcast indexing method, i.e., $ExpIn$, we developed a simulation model where several clients (50) access the data served by a server through a broadcast channel. The dataset size varies from 256 to 2048 items. We assume that the server has prior knowledge of access profiles or it employs statistical methods to estimate access statistics [17].

A. Simulation setup

For the evaluation of the method we developed a system where a node (client) requests data according to a probability distribution. Flat broadcast is employed to broadcast the data items. The data are served by a server through a broadcast channel. After the client acquires the desired data it waits for an exponential distributed time interval with mean time T before it generates another request. Thus, it is a blocking client. For every system configuration, we run 50 experiments of 5000 queries with different popular items, and we compute the average access latency and average tuning time for every experiment. We implemented a Zipfian model for item popularity distribution with parameter θ:

$$p_x = \frac{(1/x)^\theta}{\sum_{x=1}^{n}(1/x)^\theta} \qquad 1 \leq x \leq n \tag{1}$$

where θ controls the *skewness* of the access distribution for n items. For $\theta = 0$ the Zipfian reduces to the uniform distribution, whereas larger values of θ derive increasingly skewer distributions. Using this formula, we can derive relative popularities for items. As performance measures, we used the tuning time and access latency. It has to be noticed here, that time is measured according to the internal simulator clock and does not correspond to actual seconds; it only the relative timing performance of the contestants that counts and not the absolute time difference (for instance, in Figure 6 we see that the actual difference in tuning time on the leftmost plot is $5.25 - 4.35 = 0.8$, but the relative percentage is $\frac{0.9}{5.2}\% = 17\%$).

B. Energy consumption model

We assume two states of energy consumption: doze mode and active mode. E_s describes the amount of energy consumption in an energy state s per unit time. The average energy consumption can be measured by the amount of unit energy in a given time. It simulates the amount of time that a client remains tuned into the broadcast channel. The result of a query processing contains unnecessary active mode time units, because the client has to follow more pointers in order to retrieve the desired data. Note that the frequent alternation between the active and the doze by turning “on” and “off” electronic circuitry may incur additional energy consumption. However, as circuit designers become more concerned about reducing energy consumption, switching energy will become less dominant and in our experiments we don’t take it into account. The energy consumption in doze mode is about 1000 times less than that in active mode [3]. In order to evaluate energy consumption we set:

$$E_{active} = 1000 \text{ and } E_{doze} = 1 \text{ energy unit.}$$

C. Experimental results

Impact of the number of pointers.
The main idea of the Distributed Skip Air Index as described in section III is to add an extra pointer in the most popular item of the last bucket. This simple feature leads to a significant improvement in mean tuning time in cases where the distribution of data queries is somewhat skewed. We sketch here the impact of the addition of more pointers, and specifically the addition of one more pointing to the second most popular item of the last bucket. The results in Figure 5 attest that indeed there are gains. The same conclusion is easily reached in cases where a second pointer is added which points to the most popular item of the $se - 1$ bucket, and so on. In the rest of the article, we do not evaluate this version of the index further, but deal with the simpler $DiSAIn$ index.

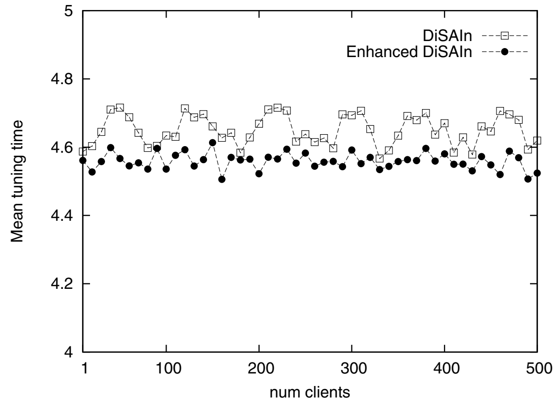


Fig. 5. Mean tuning time for $DiSAIn$ ($n = 256, \theta = 1.0$).

Impact of skewness in the access pattern.
In Figure 6 we observe the impact of skewness on the tuning time of the competing algorithms. Consistently with the theory, when the access pattern is uniform (leftmost plot), the *ExpIn* index performs as good as the $DiSAIn$. But, as the θ increases the performance gap between the two methods widens, and the $DiSAIn$ index is the improving its behavior in contrast to *ExpIn*. These observations are consistent across various numbers of items.

Performance w.r.t. the mean access time.
We evaluated the performance of the methods with respect to the access time (see Figure 7). We observed that $DiSAIn$ has no significant negative effect in mean access time. The addition of extra pointer(s) in the indexing scheme increases the size of each item leading to a total increase in mean access time. In case of one extra index to the most popular item of the last bucket, the increase in the size of each bucket is estimated according to:

$$s' = \frac{s_{ne}}{s_o + s_e}, \tag{2}$$

where s_o is the size of a data item, s_e the size of the original exponential index and s_{ne} is the size of the extra index. In a typical system of 1024 data items with $s_o = 128$ bytes and $s_e = 4$ bytes the extra index is approximately 0.4 bytes leading to a total increase of each bucket and of average access latency of 0.3%, which is really negligible. Similar results were obtained for various values of n and θ.

Performance w.r.t. the database size.
The database size plays a significant role in the performance of the exponential index. As the number of items increase, the effect that skewed data have in system performance increases also. From Figure 8, we observe that when the number of broadcasted items gets really large, then the performance gap in tuning time between the two competing indexing schemes

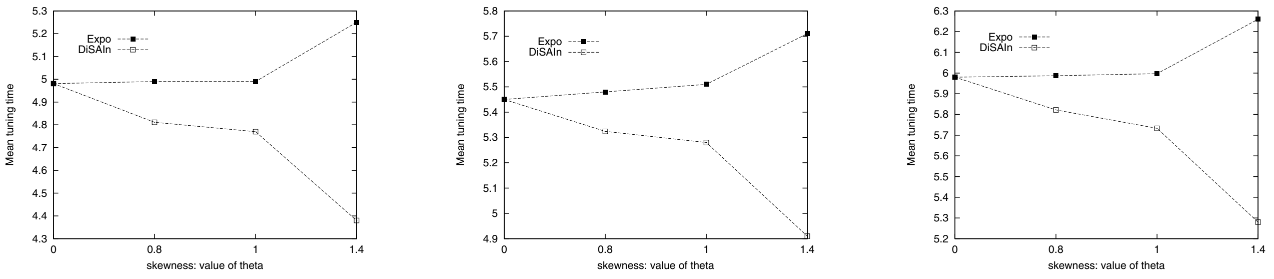


Fig. 6. Impact of skewness on mean tuning time (database size 256, 512, 1024).

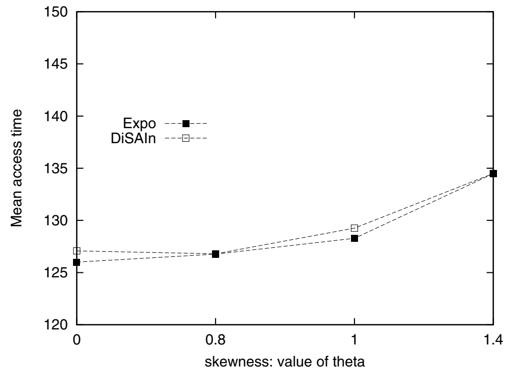


Fig. 7. Mean access latency ($n = 256, \theta = 0.5$).

broadens. This is expected since the *ExpIn* index needs to follow more pointers in order to retrieve the desired data.

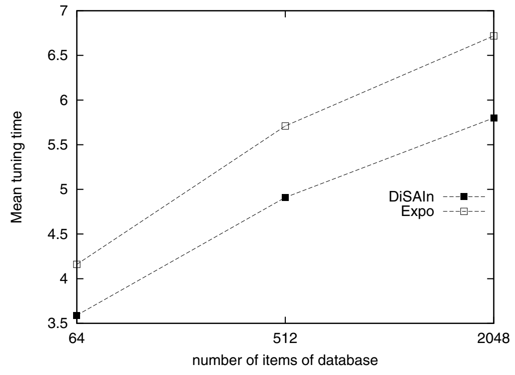


Fig. 8. Impact of number of items on mean tuning time:($\theta = 1.4$).

Performance w.r.t. energy consumption.

We evaluated the performance of the methods with respect to the energy consumption. We computed access latency and mean tuning time (Table I).

$$E_{active} = 1000 \text{ and } E_{doze} = 1 \text{ energy unit.}$$

From Table I we can compute active and doze time units and energy consumption (Table II). Active time units equal mean tuning time while doze time units are computed using the equation :

θ	*Access latency*		*Tuning time*	
	Expo	*DiSAIn*	*Expo*	*DiSAIn*
0.0	255.35	255.35	5.48	5.48
0.8	255.14	256	5.5	5.37
1.0	256	257	5.5	5.23
1.4	271	268	5.78	4.9

TABLE I
ACCESS - TUNING TIME ($n = 512, \theta = 0.0, 0.8, 1.0, 1.4$).

$$|N_{doze}| = |accesstime| - |tuningtime| \qquad (3)$$

theta	*Doze*		*Active*		*Energy*	
	Expo	*DiSAIn*	*Expo*	*DiSAIn*	*Expo*	*DiSAIn*
0.0	249.87	249.87	5.48	5.48	5729.87	5729.87
0.8	249.64	250.63	5.5	5.37	5749.64	5620.63
1.0	250.5	251.77	5.5	5.23	5750.5	5481.77
1.4	265.22	263.1	5.78	4.9	6045.22	5163.1

TABLE II
ACTIVE-DOZE TIME UNIT, ENERGY CONSUMPTION ($n = 512$).

It is obvious that *DiSAIn* is more efficient in terms of energy consumption for all experiments conducted. As the skewness of data increase *ExpIn* becomes energy inefficient while on the other hand *DiSAIn* consumes less and less energy making it the ideal choice for such environments.

V. RELEVANT WORK

Disk-based indexing (e.g., B$^+$-trees, Hashing, Skip Lists) for traditional as well as for advanced applications is a thoroughly investigated area during the past years. An adaptation of the idea of B$^+$-tree indexing in wireless environments was first described in [11], where instead of the disk addresses the leaves of the B$^+$-tree store the arrival time of each datum in the broadcast channel. Similarly, an adaptation of the traditional hash-based indexing technique in wireless environments was earlier described in [10], and later was generalized in [22]. Hybrids between the two approaches are described in [18], [24] and application of signature trees as indexing methods in reported in [14], which of course can support only equality queries. Adaptation of such indexing schemes (e.g., B$^+$-tree) to work in multiple broadcast channels are described in [8], [25]. They do not propose new schemes but simply different allocation methods for the nodes of the indexing tree. In all these works it is assumed that: a) there is a global ordering among the transmitted data, and b) the access pattern is uniform, that is, the access probability is the same for all data, which is quite unrealistic.

Deviating from the uniform access probability assumption, several works considered the effect of access skew on the design of indexing schemes. A new scheme is proposed in [19], which is a k-ary version of the basic binary Alphabetic Tree [9] over the data, whereas [23], [25] adapted the indexing method of [11] to deal with non-uniformity in access. Various methods were based on the construction of a binary or k-ary Alphabetic Tree to develop indexing schemes for multiple broadcast channels [12], [16], [28]. These methods do not provide new types of tree-structured indices, but rather a new allocation method for the tree-structured method of Alphabetic trees to the multiple channels. All these works [12], [16], [19], [23], [25] assume that: a) there is a global ordering among the transmitted data. There are only a couple of works [6], [13], which deviated from both the uniformity and global ordering assumptions. Finally, remotely related to the present work are the broadcast indexing techniques, e.g., for multidimensional data [27].

VI. Conclusions

This paper investigated the issue of indexing broadcast information under skewed access patterns. Even though there are a lot of broadcast indexing schemes in the relevant literature, our approach fills the gap by proposing a distributed index which has no root, and therefore the clients can start searching for information without waiting for a 'designated root'. The main idea of the proposed Distributed Skip Air Index is to add some pointers to the most popular items so that the clients can locate them quickly. We implemented a simulation environment to investigate the performance of the proposed scheme and presented detailed experiments that assess the superiority of the method, clearly outperforming the state-of-the-art distributed broadcast index.

References

[1] D. Aksoy and M. J. Franklin, "$R \times W$: A scheduling approach for large-scale on-demand data broadcast," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 846–860, 1999.

[2] H. Alshaer and J. M. H. Elmirghani, "Road safety based on efficient vehicular broadcast communications," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2009, pp. 1155–1160.

[3] P. V. Argade, "a high-performance low-power microprocessor," in *Proceedings of COMPCON*, 1993, pp. 88–95.

[4] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, 1999, pp. 126–134.

[5] S. S. Chawathe, "Using dead drops to improve data dissemination in very sparse equipped traffic," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2008, pp. 962–967.

[6] M.-S. Chen, K.-L. Wu, and P. Yu, "Optimizing index allocation for sequential data broadcasting in wireless mobile computing," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 1, pp. 161–173, 2003.

[7] W. Chen, R. K. Guha, T. J. Kwon, J. Lee, and Y.-Y. Hsu, "A survey and challenges in routing and data dissemination in vehicular ad hoc networks. wireless communications and mobile computing," in *Wireless Communications and Mobile Computing*, 2011, pp. 787–795.

[8] C.-H. Hsu, G. Lee, and A. L. P. Chen, "Index and data allocation on multiple broadcast channels considering data access frequencies," in *Proceedings of the Conference on Mobile Data Management (MDM)*, 2002, pp. 87–93.

[9] T. Hu and A. Tucker, "Optimal computer search trees and variable-length alphabetical codes," *SIAM Journal on Applied Mathematics*, vol. 21, no. 4, pp. 514–532, 1971.

[10] T. Imielinski, S. Viswanathan, and B. R. Badrinath, "Power efficient filtering of data on air," in *Proceedings of the Conference on Extending Data Base Technology (EDBT)*, 1994, pp. 245–258.

[11] ——, "Data on air: Organization and access," *IEEE Transactions on Knowledge and Data Engineering*, vol. 9, no. 3, pp. 353–372, 1997.

[12] S. Jung, B. Lee, and S. Pramanik, "A tree-structured index allocation mathod with replication over multiple broadcast channels in wireless environments," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 3, pp. 311–325, 2005.

[13] D. Katsaros, N. Dimokas, and Y. Manolopoulos, "Generalized indexing for energy-efficient access to partially ordered broadcast data in wireless networks," in *Proceedings of the IEEE International Database Engineering and Applications Symposium (IDEAS)*, 2006, pp. 89–96.

[14] W.-C. Lee and D. L. Lee, "Using signature techniques for information filtering in wireless and mobile environments," *Distributed and Parallel Databases*, vol. 4, no. 3, pp. 205–227, 1996.

[15] C. Liaskos, S. Petridou, G. I. Papadimitriou, P. Nicopolitidis, and A. S. Pomportsis, "On the analytical performance optimization of wireless data broadcasting," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 2, pp. 884–895, 2010.

[16] S.-C. Lo and A. L. P. Chen, "Optimal index and data allocation in multiple broadcast channels," in *Proceedings of the IEEE Conference on Data Engineering (ICDE)*, 2000, pp. 293–302.

[17] T. Sakata and J. X. Yu, "Statistical estimation of access frequencies: Problems, solutions and consistencies," *ACM Wireless Networks*, vol. 9, no. 6, pp. 647–657, 2003.

[18] A. Seifert and J.-J. Hung, "FlexInd: A flexible and parameterizable air-indexing scheme for data broadcast systems," in *Proceedings of the Conference on Extending Data Base Technology (EDBT)*, ser. Lecture Notes in Computer Science, vol. 3820, 2006, pp. 902–920.

[19] N. Shivakumar and S. Venkatasubramanian, "Efficient indexing for broadcast based wireless systems," *ACM/Baltzer Mobile Networks and Applications*, vol. 1, no. 4, pp. 433–446, 1996.

[20] N. H. Vaidya and S. Hammed, "Scheduling data broadcast in asymmetric communication environments," *ACM Wireless Networks*, vol. 5, no. 3, pp. 171–182, 1999.

[21] 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, Applications, and Services (MobiSys)*, 2004.

[22] J. Xu, W.-C. Lee, X. Tang, Q. Gao, and S. Li, "An error-resilient and tunable distributed indexing scheme for wireless data broadcast," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 3, pp. 392–404, 2006.

[23] J. Xu and K.-L. Tan, "An analysis of selective tuning schemes for nonuniform broadcast," *Data and Knowledge Engineering*, vol. 22, no. 3, pp. 319–344, 1997.

[24] X. Yang and A. Bouguettaya, "Adaptive data access in broadcast-based wireless environments," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 3, pp. 326–338, 2005.

[25] W. G. Yee and S. B. Navathe, "Efficient data access to multi-channel broadcast programs," in *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM)*, 2003, pp. 153–160.

[26] Y. Zhang, J. Zhao, and G. Cao, "Service scheduling of vehicle-roadside data access," *ACM Mobile Networks and Applications*, vol. 15, no. 1, pp. 83–96, 2010.

[27] B. Zheng, W.-C. Lee, K. C. K. Lee, D. L. Lee, and M. Shao, "A distributed spatial index for error-prone wireless data broadcast," *The VLDB Journal*, vol. 18, no. 4, pp. 959–986, 2009.

[28] J. Zhong, W. Wu, Y. Shi, and X. Gao, "Energy-efficient tree-based indexing schemes for information retrieval in wireless data broadcast," in *Proceedings of the International Conference Database Systems for Advanced Applications (DASFAA)*, 2011, pp. 335–351.