Effectively Delivering XML Information in Periodic Broadcast Environments

Yongrui Qin[1], Quan Z. Sheng[1], Muntazir Mehdi[2], Hua Wang[3], and Dong Xie[4]

[1] School of Computer Science,
The University of Adelaide, Adelaide, SA 5005, Australia
{yongrui,qsheng}@cs.adelaide.edu.au
[2] Department of Computer Science,
TU Kaiserslautern, Gottlieb-Daimler-Strasse, Kaiserslautern 67663, Germany
muntazir.75@gmail.com
[3] Department of Mathematics & Computing,
University of Southern Queensland, QLD 4350, Australia
hua.wang@usq.edu.au
[4] Department of Computer Science and Technology,
Hunan University of Humanities, Science and Technology, Loudi 417000, China
dong.xie@hotmail.com

Abstract. Existing data placement algorithms for wireless data broadcast generally make assumptions that the clients' queries are already known and the distribution of access frequencies of their queries can be obtained a priori. Unfortunately, these assumptions are not realistic in most real life applications because new mobile clients may join in anytime and clients may be reluctant to disclose their queries (due to privacy concerns). In this paper, we study the data placement problem of periodic XML data broadcast in mobile wireless environments. This is an important issue, particularly when XML becomes prevalent in today's ubiquitous Web and mobile computing devices. Taking advantage of the structured characteristics of XML data, we are able to generate effective broadcast programs based purely on XML data on the server without any knowledge of the clients' access patterns. This not only makes our work distinguished from previous studies, but also enables it to have broader applicability. We discuss structural sharing in XML data which forms the basis of our novel data placement algorithm. The proposed placement algorithm is validated through a set of experiments and the results show that our algorithm can effectively place XML data on air and significantly improve the overall access efficiency.

1 Introduction

Wireless technology has become deeply embedded in everyday life. At the end of 2011, there were 6 billion mobile subscriptions, estimated by the International Telecommunication Union (2011). That is equivalent to 87 percent of the world population, and is a huge increase from 5.4 billion in 2010 and 4.7 billion mobile subscriptions in 2009.

H. Decker et al. (Eds.): DEXA 2013, Part I, LNCS 8055, pp. 165–179, 2013.

Broadcast is one of the basic ways of information access via wireless technologies. In a wireless data broadcast system, the server broadcasts public information to all mobile devices within its transmission range via a downlink broadcast channel. Mobile clients "listen" to the downlink channel and access information of their interest directly when related information arrives. Broadcast is bandwidth efficient because all mobile clients can share the same downlink channel and retrieve data from it simultaneously. Broadcast is also energy efficient at the client ends because downloading data costs much less energy than sending data [27].

Wireless data broadcast services have been available as commercial products for many years, e.g. StarBand and Hughes Network. Recently, there has been a push for such systems from the industry and various standard bodies. For example, born out of the ITU "IMT-2000" initiative, the Third Generation Partnership Project 2 is developing Broadcast and Multicast Service in CDMA2000 Wireless IP network. Systems for Digital Audio Broadcast (DAB) and Digital Video Broadcast (DVB) are capable of delivering wireless data services. Recent news also reported that XM Satellite Radio (www.xmradio.com) and Raytheon have jointly built a communication system, known as the Mobile Enhanced Situational Awareness Network (MESA), that would use XM satellites to relay information to soldiers and emergency responders during a homeland security crisis.

On the other hand, information expressed in semi-structured formats is widespread over the past years. XML has rapidly gained popularity as a de facto standard to represent semi-structured information. Most Internet browsers provide support for XML in their newer versions and nearly all the major IT companies (e.g., Microsoft, Oracle, and IBM) have integrated XML into the software products. Delivering information in XML format is also popular in Web services and in different kinds of Publish/Subscribe systems. Consequently, XML has attracted attentions from database community recently and there has been a large body of research work focusing on XML, such as XML filtering, querying and indexing [17,26].

Combining both trends of the proliferation of mobile computing technologies and XML data, broadcasting information in XML format in a wireless environment would be a preferable way of information delivering and sharing. Consequently, the research of XML data broadcast is of great importance and in fact it has been attracting more and more research interests [20,7,24,19,18]. To further demonstrate practicability of XML data broadcast, we will present a potential application of it by detailing a real life scenario in Section 2.

There are two typical data broadcast modes: (i) *Periodic Broadcast Mode* and (ii) *On-Demand Broadcast Mode* [27]. In the periodic broadcast mode, data are periodically broadcasted on a downlink channel via which the server sends data to clients. Clients only "listen" to that channel and download data they are interested in. In the on-demand broadcast mode, clients send their queries to the server via an uplink channel and then the server considers all submitted requests and decides the contents of next broadcast cycle. In this work, we

focus on the periodic broadcast mode since it has many benefits such as saving uplink bandwidth and power at the client ends by avoiding uplink transmissions and effectively delivering information to an unlimited number of clients simultaneously.

Data placement algorithms determine what data items to be broadcasted by the server and the order of data items on wireless channels, aiming to reduce average waiting time for mobile clients. To a large extent, the data placement problem of XML data is similar to that in multi-item contexts [25,4] where mobile clients may request multiple items each time. However, there are drawbacks of existing data placement approaches in traditional data broadcast.

Firstly, previous work on multi-item placement problems generally makes assumptions that the clients' queries are already known and the distribution of access frequencies of these queries can be obtained in advance [1,2,25,4]. For example, it is proposed to allow the clients to provide a profile of their interests to the servers [1,2], but this can lead to privacy concerns. These assumptions significantly limit the practicability of the proposed placement algorithms in real situations because: (i) new mobile clients may join in the network at anytime; and (ii) mobile users may be reluctant to disclose their queries to the server via uplink channel due to expensive communication cost and privacy concerns.

Secondly, in traditional data broadcast systems, appropriate placement can hardly be generated based only on information of data items themselves on the server. Hence, the above assumptions are inevitable for the design of data placement algorithms. Alternatively, some work applies data mining techniques to discover association rules from the history access patterns of a set of data [3]. This avoids to obtain access patterns of mobile clients on-the-fly. However, the availability of such history access patterns of mobile clients is a necessity.

By contrast, in XML data broadcast, data items (or XML documents) usually share parts of their structure. Taking structural sharing between XML documents into consideration, we are able to analyze and estimate clients' access patterns via the analysis of this structural sharing. Then we can effectively place XML data on wireless channels based purely on XML data on the server, which is important for practical usage. In summary, the main contributions of this paper can be described as follows:

- By taking advantage of the structural characteristics of XML data, we are able to generate appropriate data placement results based only on XML data on the server.
- A novel data placement algorithm which organizes XML data on air is presented.
- Extensive experiments are conducted to show the effectiveness of our proposed data placement algorithm.

The remainder of this paper is organized as follows. Section 2 describes background knowledge of this work, including an application scenario, the system model and XML similarity background. Section 3 discusses the structural sharing property of XML data and proposes a novel data placement algorithm. Section 4

presents our experimental study for evaluating the performance of the proposed data placement algorithm. Finally, Section 5 discusses related work and Section 6 gives some concluding remarks.

2 Application Scenario, System Model and XML Similarity

In this section, we first describe an application scenario. Then we show the system model of this work and introduce background knowledge of XML similarity.

2.1 Application Scenario

We use the following scenario to show potential applications of XML data broadcast in real life.

Consider a live basketball game. Information about the game and the players on the court is usually the interest of a large number of audience. In this context, data broadcast is a preferable way of delivering latest information to the audience. Meanwhile, some audience could be outside of the stadium, such as basketball fans who are watching live text information about the game via the Internet at their homes. Therefore, the game information could also be delivered via the Internet to online audience and other Web service providers who have subscribed this basketball game. Using XML format to represent game information can satisfy all these needs and realize simplicity, generality, and usability of game information at the same time.

2.2 Periodic XML Data Broadcast System Model

Fig. 1 shows the model of our wireless XML data broadcast system. The system includes an XML Data Center (the broadcast server), a broadcast program scheduler, broadcast listeners (mobile clients) and a downlink channel (the server sends information to mobile clients via it). The downlink channel can be shared by all mobile clients. But mobile clients can not send their individual queries to the server in this model as no uplink channel is available.

From the figure, we can see that the XML Data Center could be connected to the Internet and deliver information to online users, Web service providers and Publish/Subscribe systems, etc. With the use of XML format data, these different applications can be integrated seamlessly with our wireless XML data broadcast system for the purpose of sharing and delivering same information to different kinds of users.

2.3 XML Similarity

Our goal is to place XML documents on the broadcast channel based only on the information at the server side. We propose to explore relatedness between different XML documents and place documents according to the relatedness results.

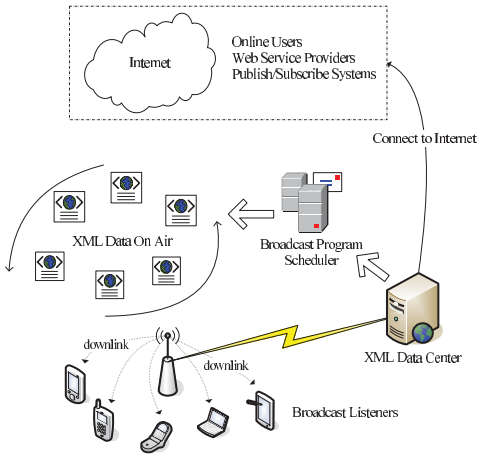


Fig. 1. A wireless XML data broadcast system

For XML documents, structural similarity is well studied and can be applied in our broadcast system as a way to calculate relatedness between documents.

Some existing work on measuring structural similarity between XML documents can be found in [23,11]. The main idea of their work is based on the concept of *path sets*. Here, a path set of an XML document contains all full paths (paths that are from root element to leaves) and their subpaths. A simple example is presented in Fig. 2. The path set of this example is: {`/player/name`, `/player/position`, `/player/nationality`, `/player/college`, `/player`, `/name`, `/position`, `/nationality`, `/college`}. We denote a path set of an XML document d as $PS(d)$.

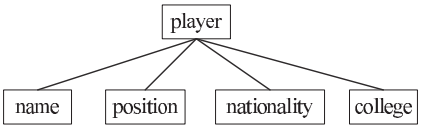


Fig. 2. An XML structure tree

Different types of measure can be adopted, such as Jaccard measure [16,10], Dice's coefficient [9] and Lian's measure [15], to measure the similarity between two XML documents d_i and d_j. The exact forms of these measures based on PS are as follows (Jaccard measure denoted as $J(d_i, d_j)$, Dice's coefficient denoted as $D(d_i, d_j)$ and Lian's measure denoted as $L(d_i, d_j)$):

$$J(d_i, d_j) = \frac{|PS(d_i) \bigcap PS(d_j)|}{|PS(d_i) \bigcup PS(d_j)|} \tag{1}$$

$$D(d_i, d_j) = \frac{2 \cdot |PS(d_i) \bigcap PS(d_j)|}{|PS(d_i)| + |PS(d_j)|} \tag{2}$$

$$L(d_i, d_j) = \frac{|PS(d_i) \bigcap PS(d_j)|}{max\{|PS(d_i)|, |PS(d_j)|\}} \tag{3}$$

From the above definitions, we can see that both Jaccard measure and Dice's coefficient give more weights on the total structural information of two comparing documents while Lian's measure emphasizes more on the difference of these documents. All three measures vary in interval $[0, 1]$. If $PS(d_i) = PS(d_j)$, we have $J(d_i, d_j) = D(d_i, d_j) = L(d_i, d_j) = 1$. Clearly, the larger the values of these measures are, the more structural sharing the two XML documents have.

3 Data Placement Algorithm

In this section, we introduce our data placement algorithm for periodic XML data broadcast. We first discuss the structural sharing property of XML data which we use to estimate the potential access patterns of mobile clients, i.e., the probability of accessing a small set of similar XML documents simultaneously. Then we put forward a novel data placement algorithm based on it.

3.1 Structural Sharing in XML Data

In the literature, two critical metrics, namely *access time* and *tuning time*, are used to measure the system's performance [12]. Data placement mainly affects access time because tuning time depends on the total content downloaded by mobile clients but not on the order of data. Hence, we use access time as our metric in this analysis. In periodic broadcast, queries are used to describe the interests of mobile clients and help mobile clients to skip irrelevant data on air, but they are not actually submitted to the broadcast server.

Intuitively, for any two given XML documents, we can utilize one of the three similarity measures described in Section 2.3 to calculate the similarity between them and the similarity results can be used to approximate the probability that a specific query is matched with both documents at the same time. For example, if two XML elements are under structurally similar paths, then it is more likely that either both elements or none satisfy a given query [23]. In fact, query issuers hardly have thorough knowledge about the broadcasted content and XPath queries usually contain * and // which would match similar structure. Therefore, if two XML documents are with larger structural similarity, i.e. d_1 and d_2, then they would have a higher probability to be required simultaneously. However, there are still three other cases to be considered, such as requiring d_1 but not d_2, requiring d_2 but not d_1 and requiring neither of d_1 and d_2. Therefore, the

Table 1. Matching Cases for Document d_1 and d_2 in a document set $\mathcal{D}$

Case	Probability	Effect on AT_{exp}
Matched both d_1, d_2	$Pr(d_1 \bigcap d_2)$	Positive
Matched none of d_1, d_2	$1 - Pr(d_1 \bigcup d_2)$	Positive
Matched d_1, but not d_2	$Pr(d_1 - d_2)$	Negative
Matched d_2, but not d_1	$Pr(d_2 - d_1)$	Negative

above similarity measures consider only successful match probabilities of both XML documents but do not consider unsuccessful match probabilities of them.

Nonetheless, unsuccessful match cases have effects on the expected access time as well (but the query may still be satisfied by other documents). In order to have better access efficiency, the distance between any two documents required by the same query should be as less uniform as possible on air. Based on this, we can infer that in the above example, cases of required d_1 but not d_2 and required d_2 but not d_1 are likely to generate more uniform distances while other two cases (required both documents or neither) are likely to have less uniform distances. Observing this, we define a new similarity measure called *Cohesion* to give a more accurate estimation of access patterns of mobile clients in the following.

Note that, for any query q requiring at least one of the documents in $\mathcal{D}$, q must match some paths in $PS(\mathcal{D})$ and it has a probability of $\frac{|PS(d)|}{|PS(\mathcal{D})|}$ to match d. If a query q fails to match any document in $\mathcal{D}$, the issuer of q will not be waiting the result to be broadcasted. Hence, we only consider satisfied queries (this means at least one document is matched) in this work.

Now suppose we have a set of n XML documents $\mathcal{D} = \{d_1, d_2, \ldots, d_n\}$ on the server, we can approximate access probability of any document d for queries which successfully match at least one document in set $\mathcal{D}$ as follows:

$$Pr(d) = \frac{|PS(d)|}{|PS(\mathcal{D})|} \tag{4}$$

and for any i, j $(1 \leq i, j \leq n)$

$$Pr(d_i - d_j) = \frac{|PS(d_i) - PS(d_j)|}{|PS(\mathcal{D})|} \tag{5}$$

Here, $PS(\mathcal{D}) = \bigcup_{i=1}^{n} PS(d_i)$.

There would be many different matching cases for a given set $\mathcal{D}$. Take two XML documents d_1 and d_2 in $\mathcal{D}$ as an example. As mentioned previously, there would be four cases of matching of them and the probability of each case is shown in Table 1. In this table, we also include positive and negative effects on the expected access time (AT_{exp}) for each case.

Based on Table 1, we define Cohesion $C(d_i, d_j)$ of XML documents d_i and d_j as follows:

$$C(d_i, d_j) = \frac{Pr(d_i \bigcap d_j) \cdot (1 - Pr(d_i \bigcup d_j))}{max\{Pr(d_i - d_j), Pr(d_j - d_i)\}} \tag{6}$$

Here d_i and d_j are both in set $\mathcal{D}$. It is easy to see that $C(d_i, d_j) = C(d_j, d_i)$. According to Equation (4), Equation (5) and Equation (6), we can calculate $C(d_i, d_j)$ after finding path sets of d_i, d_j and $\mathcal{D}$. Cohesion values can vary in a wide range which exceeds interval $[0, 1]$. Strictly speaking, Cohesion values only vary in interval $[0, \frac{|PS(\mathcal{D})|}{4}]$ given that $C(d_i, d_j) = \frac{|PS(\mathcal{D})|}{4}$ when $PS(d_i) = PS(d_j)$. The lower bound 0 is trivial. In order to obtain the upper bound, we only consider cases that have $PS(d_i) \neq PS(d_j)$, from which we can infer that $max\{|PS(d_i - d_j)|, |PS(d_j - d_i)|\} \geq 1$. Without loss of generality, let $|PS(d_i)| \geq |PS(d_j)|$, according to Equation (4) and Equation (5), we can rewrite Equation (6) as follows:

$$\begin{aligned} C(d_i, d_j) &\leq \frac{\frac{|PS(d_i \bigcap d_j)|}{|PS(\mathcal{D})|} \cdot (1 - \frac{|PS(d_i \bigcup d_j)|}{|PS(\mathcal{D})|})}{\frac{1}{|PS(\mathcal{D})|}} \\ &< \frac{-(|PS(d_i)| - \frac{|PS(\mathcal{D})|}{2})^2 + \frac{|PS(D)|^2}{4}}{|PS(\mathcal{D})|} \\ &\leq \frac{|PS(\mathcal{D})|}{4} \end{aligned}$$

Then the above result gives the upper bound of Cohesion $C(d_i, d_j)$. Now we can normalize Cohesion values to interval $[0, 1]$ in the following:

$$C'(d_i, d_j) = \begin{cases} \frac{4 \cdot C(d_i, d_j)}{|PS(\mathcal{D})|} & PS(d_i) \neq PS(d_j) \\ 1 & PS(d_i) = PS(d_j) \end{cases} \tag{7}$$

We can also infer that $C'(d_i, d_j) = 1$ if and only if $PS(d_i) = PS(d_j)$. Similar to other three similarity measures, the larger the value of Cohesion is, the more structural sharing the two comparing XML documents have.

3.2 The Data Placement Algorithm

Based on the discussion of structural sharing in XML data, we can generate a broadcast program for periodic data broadcast in a greedy way. From previous discussions, we can see that the more the structural sharing of two XML documents is, the larger probability of matching both XML documents simultaneously. As a result, our Greedy Data Placement Algorithm (GDPA) places XML documents with most structural sharing together first as an initial broadcast program. Then it progressively appends other XML documents to the broadcast program in a descendant order of structural sharing. Detailed steps of GDPA are shown in Algorithm 1 and Algorithm 2.

Algorithm 1. Initialize structural sharing matrix $S[n][n]$

Input: A set of XML documents $\mathcal{D} : \{d_1, d_2, ..., d_n\}$
Output: Structural sharing matrix $S[n][n]$
1. create matrix $S[n][n]$
2. **for** each document d in $\mathcal{D}$ **do**
3. compute $PS(d)$
4. **end for**
5. **for** each pair of documents $< d_i, d_j >$ in $\mathcal{D}$ $(i < j)$ **do**
6. $S[i][j] \Leftarrow$ structural sharing between d_i and d_j
7. $S[j][i] \Leftarrow S[i][j]$
8. **end for**

Algorithm 2. GDPA

Input: Structural sharing matrix $S[n][n]$
Output: A broadcast program σ for $\mathcal{D}$
1. $\sigma \Leftarrow$ empty sequence
2. select a pair of documents $< d_i, d_j >$ with maximum value $S[i][j]$ in matrix $S[n][n]$
3. **if** Length of $d_i <=$ Length of d_j **then**
4. add $< d_i, d_j >$ into σ
5. **else**
6. add $< d_j, d_i >$ into σ
7. **end if**
8. $\mathcal{D}' \Leftarrow \mathcal{D} - d_i - d_j$
9. **while** $\mathcal{D}'$ is not empty **do**
10. $d_{head} \Leftarrow$ the first document in σ
11. select a pair of documents $< d_{i_{max}}, d_{head} >$ with maximum value $S[i_{max}][head]$ $(d_{i_{max}} \in \mathcal{D}')$
12. $d_{rear} \Leftarrow$ the last document in σ
13. select a pair of documents $< d_{j_{max}}, d_{rear} >$ with maximum value $S[j_{max}][rear]$ $(d_{j_{max}} \in \mathcal{D}')$
14. **if** $S[i_{max}][head] >= S[j_{max}][rear]$ **then**
15. append $d_{i_{max}}$ into σ from head
16. $\mathcal{D}' \Leftarrow \mathcal{D}' - d_{i_{max}}$
17. **else**
18. append $d_{i_{max}}$ into σ from rear
19. $\mathcal{D}' \Leftarrow \mathcal{D}' - d_{j_{max}}$
20. **end if**
21. **end while**

Algorithm 1 initializes a structural sharing matrix $S[n][n]$ for n XML documents on the broadcast server. Note that, all four similarity measures defined in subsection 2.3 and 3.1 can be used in Algorithm 1 to compute structural sharing between two documents (Line 6). All of them are symmetric which means for any one of these measures, we must have $S[j][i] = S[i][j]$. Also we have $J(d_i, d_j) = D(d_i, d_j) = L(d_i, d_j) = C'(d_i, d_j) = 1$ if $i = j$. Therefore, we only need to calculate matrix S for entries $S[i][j]$ where $i < j$.

Based on matrix S, Algorithm 2 finds the pair of XML documents with maximum structural sharing and adds them into the initial empty broadcast program σ (Line 2). The sequence of the first pair of XML documents are placed according to the ascendant order of document lengths (Line 3 to 7). Then Algorithm 2 appends the XML document with maximum structural sharing to the head document d_{head} or the rear document d_{rear} of σ. If the maximum structural sharing is derived between document d and document d_{head}, d will be appended into σ from head; otherwise, d will be appended into σ from rear. This process will be repeated until all XML documents are placed into σ (Line 9 to 21).

Regarding the computing complexity of Algorithm 2, the main task of the scheduling is performed from Line 9 to 21. The whole 'while' block has at most n loops. Within this block, Line 11 takes $O(n)$ time. It is similar at Line 13, which also takes $O(n)$ time. Hence the time complexity of the whole 'while' block is $O(n^2 + n^2)$. Meanwhile, the complexity of Line 2 is $O(n^2)$. As a result, the complexity of Algorithm 2 is $O(3n^2)$.

4 Experiments

Since this is the first work that determines broadcast schedules based only on XML data on the server without any knowledge of the clients' access patterns, we compare our algorithm with a common random data placement algorithm (RDPA) and show its efficiency in terms of access time, which is a common measure of performance in data broadcasts. We have not compared tuning time as the comparing data placement algorithms would not affect it.

4.1 Experimental Setup

The experiments are run on three data sets each with 250 XML documents defined by News Industry Text Format (NITF) DTD [13], which is published for news copy production, press releases, and Web-based news organizations. The average depth of the three document sets is between 6 and 8 while the maximum depth is 20.

There are three data sets in the experiments, which are $DS1$, $DS2$ and $DS3$. Data in $DS1$ can be well clustered into 6 clusters. Moreover, for any two documents d_i, d_j in two different clusters of $DS1$, the minimum similarity values, the maximum similarity values and the average similarity values of all four measures (normalized Cohesion is adopted here) are shown in Table 2. We can see that all clusters are quite different from each other and share very little structural information. Data in $DS2$ are miscellaneous. Documents in $DS2$ cannot be classified into fine clusters. Data in $DS3$ are a mix of well-clustered data and miscellaneous data, which include 125 XML documents from $DS1$ and 125 XML documents from $DS2$.

In the experiments, XPath queries are generated using the generator developed by [8]. Queries are allowed to repeat. The generator provides several parameters to generate different types of XPath queries, such as query depth,

Table 2. Similarity between clusters in $DS1$

Measure	Similarity		
	Minimum	Maximum	Average
Jaccard	0.0097	0.1102	0.0435
Dice	0.0049	0.0583	0.0225
Lian	0.0057	0.1039	0.0345
Cohesion	0.0229	0.4620	0.1457

Table 3. Workload Parameters for the Experiments

Parameter	Range	Default	Description
$PROB$	5% to 30%	10%	probability(* and //)
QIR	0.1 to 5	1	query incoming rate
MQD	5 to 8	7	maximum query depth

probability of * and // and the maximum depth of generated XPath queries. The probability of * and // appearing in each query's step is between 5% and 30% (denoted $PROB$, and the default value is 10%). Note that, Query Incoming Rate (denoted QIR) means the number of newly issued queries from mobile clients in a unit of time (these queries are only locally issued for data retrieval purpose and are not sent to the broadcast server). We measure this unit of time by the time that mobile wireless system takes to broadcast a block of 1024-byte XML data. The maximum depth of generated XPath queries (denoted MQD) is between 5 and 8. Table 3 shows details of the parameters in the experiments.

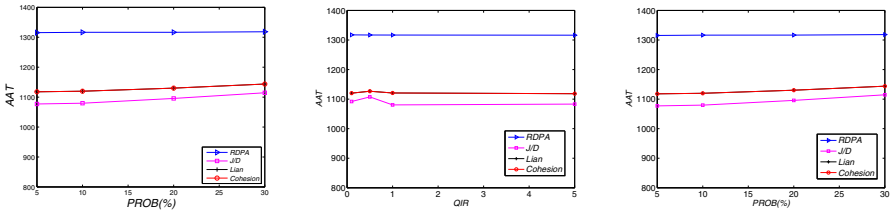
The random data placement algorithm (RDPA) is compared with GDPA (using all four similarity measures defined in Equations (1), (2), (3) and (7)). In RDPA, the server broadcasts XML documents in a random order.

We implement both RDPA and GDPA on Java Platform Standard Edition 6 running on Windows 7 Enterprise, 64-bit Operating System. All our experiments are obtained by running 30 consecutive broadcast cycles. When we vary $PROB$, we set QIR and MQD to their default values. When we vary QIR, we set $PROB$ and MQD to their default values. Similarly, when we vary MQD, we set $PROB$ and QIR to their default values.

Regarding air indexing and index distribution strategy, in our experiments, we adopt Compact Index (CI) [24] as our index structure and $(1, m)$ index scheme [12] as our index distribution strategy. This is because CI is the state-of-the-art indexing technique for XML data broadcast and $(1, m)$ index scheme is the most popular index distribution strategy for traditional periodic data broadcast. More details can be found in [24] and [12].

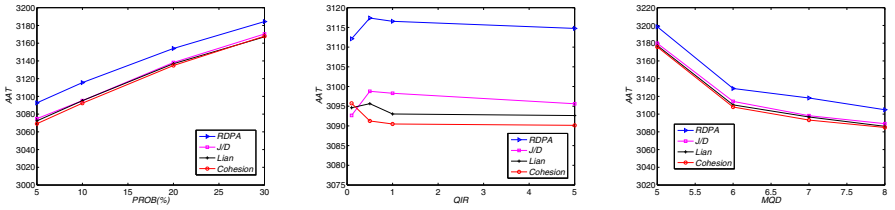
4.2 Performance of GDPA

Our experimental results are shown in Fig. 3, Fig. 4 and Fig. 5. Average access time (AAT) is our performance metric. Also we only consider AAT for



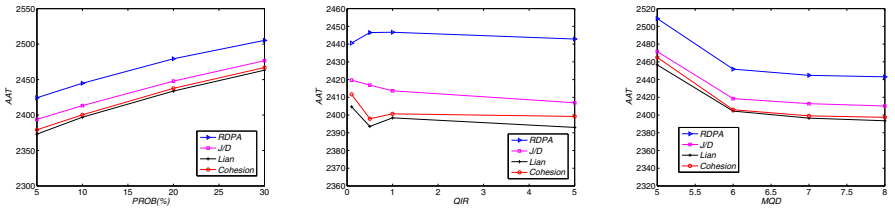
(a) Varying $PROB$ (b) Varying QIR (c) Varying MQD

Fig. 3. Evaluating AAT Performance on $DS1$: well-clustered data set



(a) Varying $PROB$ (b) Varying QIR (c) Varying MQD

Fig. 4. Evaluating AAT Performance on $DS2$: miscellaneous data set



(a) Varying $PROB$ (b) Varying QIR (c) Varying MQD

Fig. 5. Evaluating AAT Performance on $DS3$: a mixed set of well-clustered data and miscellaneous data

all successful matched queries and abandon unsuccessful matched queries. The main reason for this is that, AAT of unsuccessful queries is determined by index distribution but not by data placement results (more details about this can be found in [12]). Note that, GDPA can be implemented with four different similarity measures defined in Section 3, which are Jaccard measure, Dice's coefficient, Lian's measure and our proposed Cohesion. Through our experiments, Jaccard measure and Dice's coefficient always yield the same results. Therefore, we denote GDPA implemented with them as *J/D* method in all figures. Meanwhile, we denote GDPA implemented with Lian's measure as *Lian* method and denote GDPA implemented with Cohesion as *Cohesion* method.

Fig. 3 shows the results on $DS1$. From the figure we can see that all GDPA methods outperform RDPA significantly. Specifically, J/D method achieves the

best results while Lian method and Cohesion method provides similar results. This indicates that J/D method better fits well-clustered data. In Fig. 3(a), GDPA methods become slightly worse when $PROB$ increases. Since $DS1$ is well-clustered, most queries only require documents in the same clusters. Thus $PROB$ has less effect on AAT. In Fig. 3(b), when QIR increases, J/D method becomes slightly better. This indicates that J/D method can achieve better scalability than other methods when accessing well-clustered data. Fig. 3(c) shows that all GDPA methods remain stable as MQD increases. It is interesting to note that for RDPA, AAT always remains stable.

Fig. 4 shows the results on $DS2$. From the figure we can see that all GDPA methods achieve better performance when compared with RDPA. Specifically, Cohesion method achieves the best results while J/D method achieves the worst results among GDPA methods. This indicates that Cohesion method better fits miscellaneous data. In Fig. 4(a), both GDPA methods and RDPA become worse when $PROB$ increases. It is clear that $PROB$ has more effect on AAT for miscellaneous data. In Fig. 4(b), when QIR increases from 0.1 to 0.5, GDPA methods J/D and Lian together with RDPA become worse while Cohesion method still becomes better. After that, when QIR increases, all methods become slightly better. This shows that Cohesion method can achieve best scalability when accessing miscellaneous data.

Fig. 5 shows the results on $DS3$. Similarly, all GDPA methods achieve better performance when compared with RDPA. Specifically, Lian method achieves the best results while J/D method provides the worst results among GDPA methods. This shows that Lian method better fits hybrid data. However, Cohesion method achieves very similar performance of Lian method. In Fig. 5(a), both GDPA methods and RDPA become worse when $PROB$ increases. $PROB$ has more effect on AAT for hybrid data. In Fig. 5(b), when QIR increases, all GDPA methods become slightly better and still Lian method provides the best results.

To sum up, GDPA methods always achieve better AAT when compared with RDPA. When accessing well-clustered data, J/D method achieves the best performance. When accessing miscellaneous data, Cohesion method provides the best performance and finally when accessing hybrid data, Lian method shows the best performance.

5 Related Work

Multi-item data placement problem is related to the data placement problem of XML data which is the focus of our work. It is proved to be a NP-Complete problem [6].

Existing data placement methods for processing multi-item queries in periodic broadcast[5,14,3] generally makes assumptions that the clients' queries are already known and the distribution of access frequencies of these queries can be obtained in advance. However, these assumptions are not true for most applications in real life because the demand is either not known or it may be costly to collect the demand information.

Multi-item data placement problem in on-demand broadcast mode has also attracted lots of interests [25,22]. These approaches are in pure on-demand broadcast mode and strictly require that mobile clients submit their queries to the server for desired data. Otherwise, the server will not broadcast related data on air. This is because the server filters and schedules data solely based on submitted queries. However, frequent use of uplink channel leads to high communication cost via uplink channel, which can shorten battery life of mobile clients dramatically.

The most related work is proposed in [21] where the broadcast schedules are generated based on clustering results of XML data on the server. However, when finding the optimal clustering result, the clustering process requires manually specifying the number of clusters and has to compare different clustering results based on clients' query distribution, which differs from our work in this paper.

6 Conclusion

In this paper, we have studied the data placement problem of periodic XML data broadcast in mobile wireless environments. Taking advantage of the structured characteristics of XML data, we are able to generate effective broadcast programs based only on XML data on the server without any knowledge of the clients' access patterns. This not only makes our work distinguished from previous studies, but also enables it to have broader applicability. Our experiments demonstrated that the proposed algorithm could improve access efficiency and achieve better scalability.

In the future, we plan to further improve system's performance by investigating the insights of structural sharing among XML documents. For example, we may consider details on how to measure structural sharing distribution in an XML document set, how the distribution affects the expected access time of queries and how to choose a similarity measure based on structural sharing distribution in a set of XML documents.

References

1. Acharya, S., Alonso, R., Franklin, M.J., Zdonik, S.B.: Broadcast Disks: Data Management for Asymmetric Communications Environments. In: SIGMOD, pp. 199–210 (1995)
2. Acharya, S., Franklin, M.J., Zdonik, S.B.: Balancing Push and Pull for Data Broadcast. In: SIGMOD Conference, pp. 183–194 (1997)
3. Chang, Y.I., Hsieh, W.H.: An Efficient Scheduling Method for Query-Set-Based Broadcasting in Mobile Environments. In: ICDCS Workshops, pp. 478–483 (2004)
4. Chen, J., Lee, V.C.S., Liu, K.: On the Performance of Real-time Multi-item Request Scheduling in Data Broadcast Environments. Journal of Systems and Software 83(8), 1337–1345 (2010)
5. Chung, Y.D., Kim, M.H.: QEM: A Scheduling Method for Wireless Broadcast Data. In: DASFAA, pp. 135–142 (1999)

6. Chung, Y.D., Kim, M.H.: Effective Data Placement for Wireless Broadcast. Distributed and Parallel Databases 9(2), 133–150 (2001)
7. Chung, Y.D., Lee, J.Y.: An Indexing Method for Wireless Broadcast XML Data. Inf. Sci. 177(9), 1931–1953 (2007)
8. Diao, Y., Altinel, M., Franklin, M.J., Zhang, H., Fischer, P.M.: Path Sharing and Predicate Evaluation for High-Performance XML Filtering. ACM Trans. Database Syst. 28(4), 467–516 (2003)
9. Dice, L.R.: Measures of the Amount of Ecologic Association Between Species. Ecology 26(3), 297–302 (1945)
10. Ganesan, P., Garcia-Molina, H., Widom, J.: Exploiting Hierarchical Domain Structure to Compute Similarity. ACM Trans. Inf. Syst. 21(1), 64–93 (2003)
11. Helmer, S.: Measuring the Structural Similarity of Semistructured Documents Using Entropy. In: VLDB, pp. 1022–1032 (2007)
12. Imielinski, T., Viswanathan, S., Badrinath, B.R.: Data on Air: Organization and Access. IEEE Trans. Knowl. Data Eng. 9(3), 353–372 (1997)
13. IPTC: International Press Telecommunications Council, News Industry Text Format (NITF), `http://www.nitf.org`
14. Lee, G., Yeh, M.S., Lo, S.C., Chen, A.L.P.: A Strategy for Efficient Access of Multiple Data Items in Mobile Environments. In: MDM, pp. 71–78 (2002)
15. Lian, W., Cheung, D.W.L., Mamoulis, N., Yiu, S.M.: An Efficient and Scalable Algorithm for Clustering XML Documents by Structure. IEEE Trans. Knowl. Data Eng. 16(1), 82–96 (2004)
16. Lin, D.: An Information-Theoretic Definition of Similarity. In: ICML, pp. 296–304 (1998)
17. Miliaraki, I., Koubarakis, M.: FoXtrot: Distributed structural and value XML filtering. TWEB 6(3), 12 (2012)
18. Park, C.S., Park, J.P., Chung, Y.D.: PrefixSummary: A Directory Structure for Selective Probing on Wireless Stream of Heterogeneous XML Data. IEICE Transactions 95-D(5), 1427–1435 (2012)
19. Park, J.P., Park, C.S., Chung, Y.D.: Energy and Latency Efficient Access of Wireless XML Stream. J. Database Manag. 21(1), 58–79 (2010)
20. Park, S.-H., Choi, J.-H., Lee, S.: An Effective, Efficient XML Data Broadcasting Method in a Mobile Wireless Network. In: Bressan, S., Küng, J., Wagner, R. (eds.) DEXA 2006. LNCS, vol. 4080, pp. 358–367. Springer, Heidelberg (2006)
21. Qin, Y., Wang, H., Sun, L.: Cluster-Based Scheduling Algorithm for Periodic XML Data Broadcast in Wireless Environments. In: AINA Workshops, pp. 855–860 (2011)
22. Qin, Y., Wang, H., Xiao, J.: Effective Scheduling Algorithm for On-Demand XML Data Broadcasts in Wireless Environments. In: ADC, pp. 95–102 (2011)
23. Rafiei, D., Moise, D.L., Sun, D.: Finding Syntactic Similarities Between XML Documents. In: DEXA Workshops, pp. 512–516 (2006)
24. Sun, W., Yu, P., Qin, Y., Zhang, Z., Zheng, B.: Two-Tier Air Indexing for On-Demand XML Data Broadcast. In: ICDCS, pp. 199–206 (2009)
25. Sun, W., Zhang, Z., Yu, P., Qin, Y.: Efficient Data Scheduling for Multi-item Queries in On-Demand Broadcast. In: EUC (1), pp. 499–505 (2008)
26. Vagena, Z., Moro, M.M., Tsotras, V.J.: RoXSum: Leveraging Data Aggregation and Batch Processing for XML Routing. In: ICDE, pp. 1466–1470 (2007)
27. Xu, J., Lee, D.L., Hu, Q., Lee, W.C.: Handbook of Wireless Networks and Mobile Computing, pp. 243–265. John Wiley & Sons, Inc. (2002)