# Effective Scheduling Algorithm for On-Demand XML Data Broadcasts in Wireless Environments

**Yongrui Qin**[1] **Hua Wang**[1] **Jitian Xiao**[2]

[1]Department of Mathematics & Computing
University of Southern Queensland, Australia,
Email: yongrui.qin@usq.edu.au, wang@usq.edu.au

[2] School of Computer and Security Science
Edith Cowan University, Australia,
Email: j.xiao@ecu.edu.au

**Abstract**

The organization of data on wireless channels, which aims to reduce the access time of mobile clients, is a key problem in data broadcasts. Many scheduling algorithms have been designed to organize flat data on air. However, how to effectively schedule semi-structured information such as XML data on wireless channels is still a challenge. In this paper, we firstly propose a novel method to greatly reduce the tuning time by splitting query results into XML snippets and to achieve better access efficiency by combining similar ones. Then we analyze the data broadcast scheduling problem of on-demand XML data broadcasts and define the efficiency of a data item. Based on the definition, a Least Efficient Last (LEL) scheduling algorithm is also devised to effectively organize XML data on wireless channels. Finally, we study the performance of our algorithms through extensive experiments. The results show that our scheduling algorithms can reduce both access time and tuning time significantly when compared with existing work.

*Keywords:* wireless environment, data broadcast, on-demand, XML, multi-item, scheduling algorithm

## 1 Introduction

With the rapid development of wireless network technologies, users with mobile devices (such as palmtops, PDAs, WAP phones and so on) can access a large amount of information at anytime from anywhere. Over the past few years, mobility and portability have created an entire new class of applications. For example, information services, including news, stock quotes, airline schedules, weather report and traffic information, are becoming more and more popular and helpful. Logically, information access via these wireless technologies can be classified into two basic ways: point-to-point access and broadcast (Imielinski et al. 1997, Xu et al. 2002).

Point-to-point access employs a pull-based approach where a mobile client initiates a query to the server which in turn processes the query and returns the result to the client over a point-to-point channel. It is suitable for lightly loaded systems in which wireless channels and server processing capacity is not severely contended.

On the other hand, data broadcast is an efficient way for public information delivery to a large number of mobile users. It offers great scalability, good power consumption, and efficient bandwidth utilization (Imielinski et al. 1997, Xu et al. 2002). In addition, it allows an arbitrary number of clients to access data simultaneously and thus is particularly suitable for heavily loaded systems. Recently, there has been a push for such systems from the industry and various standard bodies. For example, born out of the International Telecommunication Union's (ITU) International Mobile Telecommunications "IMT-2000" initiative, the Third Generation Partnership Project 2 (3GPP2 2007) is developing Broadcast and Multicast Service in CDMA2000 Wireless IP network.

There are two typical broadcast modes for data broadcast (Xu et al. 2002): 1) Broadcasting Mode. Data is periodically broadcast on the downlink channel. Clients only "listen" to that channel and download data they are interested in; 2) On-Demand Mode. The clients send their requests to the server through uplink channel and the server considers all pending requests to decide the contents of next broadcast cycle. In this paper, we focus on on-demand data broadcasts.

Access efficiency and power conservation are two important issues in wireless data broadcast system since mobile clients are typically powered by batteries with limited capacity. Accordingly, two critical metrics, access time and tuning time are used to measure the system's performance (Imielinski et al. 1997, Xu et al. 2002). Access time refers to the time elapsed from the moment a query is issued to the moment it is answered while tuning time refers to the time a mobile client stays in *active mode* to receive the requested information.

Aiming at reducing tuning time, Imielinski et al. (1997), Xu et al. (2002), Lee & Zheng (2005) study air indexing techniques. They introduce some auxiliary data structures in broadcast to indicate the arrival time of each data item on a wireless channel. As a result, mobile clients know the arrival time of the requested data items in advance and can switch to the energy-saving mode (*doze mode*) during waiting. Therefore, the advantage of air index is reducing tuning time and thus a longer battery life can be attained.

Broadcast schedule determines what data items to be broadcast by the server and also the order of data items on wireless channels. Acharya et al. (1995), Acharya & Muthukrishnan (1998), Aksoy & Franklin (1999), Sun et al. (2003), Huang et al. (2010 (in print)

investigate scheduling techniques which aim to reduce access time. These studies are under the premise that each user query requires only one data item. Other work studies the multi-item queries scheduling problems (Chung & Kim 1999, Lee et al. 2002, Sun et al. 2008).

Besides the traditional flat data information, such as records in relational databases, more and more information are described in semi-structured format over the past few years. XML has rapidly gained popularity as a standard to represent semi-structured information, and is also considered an effective format for data transmission and exchange.

**Motivation** To a large extent, scheduling XML data is similar to multi-item scheduling problem. However, previous multi-item scheduling algorithms mainly take advantage of access frequencies of data items and different queried result sets containing multiple items (Chung & Kim 1999, Lee et al. 2002, Sun et al. 2008) but not the sizes of data items as well. In contrast, data items (or XML files) have a variety of lengths in XML data broadcasts and thus the lengths of data items should be taken into account. Furthermore, in traditional data broadcasts, data items are usually considered atomic, but in XML data broadcasts, the data items are no longer atomic. Therefore, when scheduling XML data, we should consider not only the order of data items on the wireless channel, but also the structured characteristics of XML data. To the best of our knowledge, little scheduling work on data broadcasts has addressed scheduling problems for non-atomic data items.

In this paper, we focus on the scheduling problem of on-demand XML data broadcasts. Firstly, we discuss two naive schemes on how to match XML data against mobile users' queries. On the basis of these two schemes, we make use of the structured characteristics of XML data and propose to only broadcast as less redundant information as possible to reduce tuning time. We also put forward a more practical scheme which combines similar queried results to achieve better performance of access time with a little overhead of tuning time. Then we analyze the scheduling problem of on-demand XML data broadcasts and devise an improved scheduling algorithm to achieve better access efficiency. In summary, the main contributions of this paper are:

- We propose to pre-process XML data before broadcast to reduce both tuning time and access time. Taking advantage of the structured characteristics of XML data, we discuss two naive schemes on how to match XML data with mobile users' queries. Then a more practical scheme is put forward to reduce access time with a little overhead to the optimal tuning time.
- We analyze the scheduling problem of on-demand XML data broadcasts and give a formal definition of the efficiency of a data item. Based on this definition, we then devise a Least Efficient Last (LEL) scheduling algorithm to effectively schedule XML data on wireless channels. In addition, computing complexity and theoretical analysis of LEL scheduling algorithm are also given.
- We perform extensive experiments to study the effectiveness of our solutions. These experiments show that our solutions can achieve better performance when compared with existing work.

We proceed with related work in Section 2. Section 3 proposes a pre-processing technique for on-demand XML data broadcasts to reduce tuning time and to achieve better access efficiency. Section 4 analyzes the scheduling problem of XML data in on-demand broadcasts and puts forward an improved scheduling algorithm. Section 5 presents our experimental study and evaluates the performance of the proposed approach. Finally, Section 6 concludes this paper.

## 2 Related Work

Recently, a lot of work dealing with XML data broadcast has appeared. Chung & Lee (2007), Park et al. (2005, 2006) address the performance optimization of query processing of XML streams in wireless broadcast. On the other hand, Qin et al. (2009), Sun et al. (2009) design some indexing techniques for XML data broadcasts based on existing XML indexing techniques. However, their work mainly focuses on air indexing techniques and does not discuss the scheduling problem in XML data broadcasts.

Multi-item scheduling problem is also related to the scheduling problem of XML data. Multi-item scheduling problem is proved to be a NP-Complete problem (Chung & Kim 1999). Also a scheduling method for multi-item queries called QEM is introduced, which opened up a new perspective in this field. In addition, the measure Query Distance (QD) is defined .It shows the coherence degree of a query's data set in a schedule. Chung & Kim (1999) prove it could represent the AT of the query. The basic idea of QEM is to expand the data of each query according to their access frequencies.

Lee et al. (2002) propose Modified-QEM. It loosens the restriction that the QD of previously expanded queries cannot be changed. In fact a "move" action could be executed so that the QD of the new coming query is optimized. By doing "move" at specified times, the algorithm improved the QEM performance.

Chang & Hsieh (2004) propose another algorithm called Improved QEM. It employs the association rules in data mining to discover the relationship among data items and then applies QEM method on data sets instead of queries. Improved QEM method does consider different length of data items. However, it is still QEM based and is proven to be less efficient than the method proposed by (Sun et al. 2008).

Sun et al. (2008) put forward a scheduling algorithm named LRL (short for Least Required Last) algorithm. Since the queries which will be fully satisfied[1] by the last broadcast data item have the longest access time, it is reasonable to have as fewer queries as possible to be fully satisfied by the last data item. Thus, the last broadcast data item should have the least access frequency. Up to now, LRL scheduling algorithm shows the best results when compared with other work. However, this algorithm does not make use of different sizes of data items.

## 3 Pre-processing XML Data

Consider the queried results of different mobile users' queries submitted almost at the same time. In traditional data broadcast, the queried result comprises of a set of data items, such as $d_1$, $d_2$, $d_3$ and so forth. These data items are independent of the queries which means the queries will not affect the content of the data items. In other words, a matched data item is regarded as atomic item and cannot be divided into smaller ones. The queries match the whole data item

[1] 'Fully satisfied' means that the issuers of these queries have received all data items they required.

other than some parts of its content. As a result, traditional scheduling algorithms only need to consider the placements of data items that are selected by mobile users' queries.

In contrast, the queried results of XML data are no longer atomic data items, but a chunk of structured data that satisfy users' queries. This chunk of structured data is usually part of a whole XML file. Actually, different queries may request the same XML file. However, only parts of the file match these queries and these parts might be quite different from each other. To explain it in another way, the queried results in each XML data file are dependent on users' queries that are sent to the server. Thus, the queried results should not be regarded as atomic data items (see Figure 1). If each XML file is treated as an atomic data item, which is similar to traditional data broadcast, mobile users who have submitted queries that match some parts of this XML file all need to download the whole XML file from the wireless channel. It obviously brings down the overall access efficiency and energy efficiency. Therefore, in order to enhance the broadcast system performance, a pre-processing phase is needed in on-demand XML data broadcasts. According to this finding, we first discuss two naive schemes for the pre-processing phase.

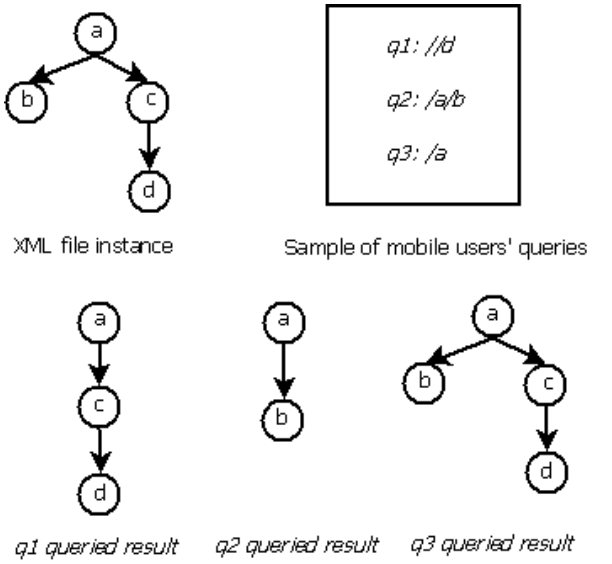



Figure 1: XML file instance, sample of mobile users' queries and the queried results

### 3.1 Two Naive Schemes

There are two naive schemes to pre-process XML data in on-demand broadcasts. One scheme follows the concept of traditional data broadcast and does not take advantage of the structured characteristics of XML data while the other scheme only broadcasts XML snippets or sub-trees those mobile users' queries exactly match. These schemes can be described as follows:

1. *Atomic Scheme*: Similar to the traditional data broadcasts, in this scheme, the server treats every XML file as an atomic item. Thus once a mobile user query matches some part of an XML file, the whole XML file will be broadcast by the server. For example, in Figure 1, since all three queries match the XML file instance, the whole file instance will be broadcast on the wireless channel and all these three queries can be satisfied by it. In this case, the total tuning time is 12, if simply in terms of nodes (4 nodes for each query).

2. *Snippet Scheme*: In this scheme, by taking the structured characteristics of XML data into consideration, the server regards XML files as non-atomic items and all XML files are dividable. The server broadcasts every XML snippet that satisfies any of the mobile users' queries. For example, in Figure 1, there are three XML snippets that satisfy the three queries respectively. All these snippets are different from each other and so the server broadcasts each of them on the wireless channel. The mobile users of each query can just download the required XML snippet and skip the other two snippets. Therefore, the total tuning time is 9 (that is 3+2+4), if in terms of nodes.

On the one hand, *Atomic scheme* is commonly used in previous work (Chung & Lee 2007, Park et al. 2005, 2006, Sun et al. 2008, 2009, Qin et al. 2009) since it results in least broadcast content on the wireless channel. On the other hand, *Snippet Scheme* is just a theoretical model and is never used in previous work. This is because although it can achieve the best tuning time for each mobile user, it can cause a lot of redundant content to be broadcast on the wireless channel. Thus, lots of mobile users have to spend longer waiting time. Clearly, in the *atomic scheme*, most of the mobile users download a large amount of redundant XML data and thus the tuning time is extreme large while in the *snippet scheme*, each mobile user only downloads the exact XML data they require and thus the tuning time is optimal. Nonetheless, the *snippet scheme* is still not the ideal one because it dose not consider the similarity among different snippets and thus better access time can still be expected. Hence, we put forward a more effective and practical scheme in the following subsection.

### 3.2 Combining Scheme

In the *combining scheme*, we further make use of the similarity of different XML snippets based on the *snippet scheme*. In this way, we can reduce the overall access time with a little tuning time overhead.

For instance, in Figure 1, since the queried results of $q_1$ and $q_3$ are very similar, we can directly use the queried result of $q_3$ to satisfy $q_1$. Thus, the users of $q_1$ need to download only one more node (in this case, node b in the XML file instance). Actually, the queried results of $q_1$ and $q_3$ can be combined into one since they are quite similar. In this example, the optimal total access time of the *snippet scheme* is 16 nodes (2 nodes waiting time for $q_2$, 5 nodes waiting time for $q_1$, 9 nodes waiting time for $q_3$), while in the *combining scheme*, the optimal total access time is 14 nodes (2 nodes waiting time for $q_2$, 6 nodes waiting time for both $q_1$ and $q_3$). Moreover, the optimal total access time for the *atomic scheme* is 15 nodes (5 nodes waiting time for all three queries). Therefore, the *combining scheme* can achieve better access efficiency when compared with the two naive schemes. Moreover, it causes only a little tuning time overhead to the optimal result in theory(the total tuning time is 10 nodes, only 1 more node than that of the *snippet scheme*, which is the optimal one). Hence, it can provide much better balance between access time and tuning time (See the below Table 1, the numbers in *italic* are the best values among three schemes).

Table 1: Comparison of three schemes for Figure 1 in terms of XML nodes

| Scheme | Access Time | Tuning Time |
|---|---|---|
| *atomic scheme* | 15 | 12 |
| *snippet scheme* | 16 | *9* |
| *combining scheme* | *14* | 10 |

In the *combining scheme*, we need to define a parameter *sim* (short for similarity) to measure the similarity of two XML snippets. Suppose that there are two XML snippets with lengths of $L_1$ and $L_2$ respectively, and the length of the combined one is $L_3$. Clearly, we have $Min(L_1, L_2) \leq Max(L_1, L_2) \leq L_3$.

**Definition 1.** The similarity of two XML snippets *sim* is defined as follows:

$$sim = \frac{L_3}{Min(L_1,L_2)}$$

Obviously, *sim* always satisfies $sim \geq 1$ and $sim \geq \frac{Max(L_1,L_2)}{Min(L_1,L_2)}$. A smaller *sim* indicates more similarity between the two XML snippets before combined while a larger *sim* indicates less similarity. If $sim = 1$, then the two XML snippets are exactly the same. Therefore, if we suppose to combine only XML snippets that are very similar to each other, we just need to combine snippets that have small *sim* values. In Section 5 we will show that combining XML snippets with *sim* values equal to or smaller than 1.4 can achieve the best overall performance.

Note that, in the *combining scheme*, the combined result is always part of the original XML file instance, which means the local structured information will remain unchanged. In the example in Figure 1, when combining the two similar XML snippets, we just use queries $q_1$ and $q_3$ together to match against the XML file instance and then we can get the combined result of these two snippets. However, the so-called combining stage is not so straightforward.

We do not have to generate separate queried results of users' queries first and then combine those results. But actually we just adopt some XML data filtering techniques to find the combined result efficiently. By using the XML data filtering techniques (Vagena et al. 2007, Diao et al. 2003), we can determine which parts of a specific XML file match some queries efficiently. Then we can calculate the *sim* after the filtering has finished by just identifying which part of the filtering result is matched the queries. After that, take $q_1$ and $q_3$ for instance, if the *sim* value indicates that the queried results of $q_1$ and $q_3$ should be combined, then the filtering result of $q_1$ and $q_3$ is directly treated as a new XML snippet. We should continue the calculations with other queried results until no *sim* value of any two XML snippets is equal to or smaller than the specified threshold of *sim* value, such as 1.4 (in this case, we can firstly exclude XML snippets with $sim \geq \frac{Max(L_1,L_2)}{Min(L_1,L_2)} > 1.4$ and then find possible combining results).

At the client side, the clients just need to download the XML data they are interested in. However, as mentioned before, in the *combining scheme*, since the XML data is made up of combined snippets, the clients may download XML data that contains other information they are not interested in but required by other clients. This is due to the combination of different queried results in a snippet but better balance between access efficiency and power conservation can be expected.

## 4 Scheduling XML Data

At this stage, the scheduling problem of on-demand XML data broadcast is quite similar to the scheduling problem in multi-item data broadcast. The scheduling problem of multi-item data broadcast is known as a NP-Complete problem (Chung & Kim 1999). Up to now, a large body of studies has been done to solve the scheduling problem of multi-item data broadcast. However, previous work mainly considers access frequencies of data items and different queried result sets containing multiple items. In XML data broadcast, the data items have a variety of lengths and thus scheduling algorithms should take it into account. Moreover, existing scheduling algorithms suggest that the server only broadcast part of the queried results of the pending queries in one broadcast cycle (or one schedule) in order to achieve better access efficiency. This idea is under the premise that the data items are all atomic and each data item may be required by different queries. However, in on-demand XML data broadcast, data items are no longer atomic and thus each broadcast cycle aims to satisfy all pending queries that were issued during the previous broadcast cycle period. Based on this observation, we analyze the new model of scheduling XML data in on-demand broadcast and then put forward an improved scheduling algorithm in the following.

### 4.1 Analysis of Scheduling On-Demand XML Data Broadcasts

Consider the broadcast scenario of on-demand XML data broadcasts in Figure 2. At time t0 and t1, queries $q_i$ and $q_j$ arrive respectively. Then a new broadcast cycle begins at time t2. This cycle aims to satisfy all the queries that were issued during the last broadcast cycle period, such as queries $q_i$, $q_j$ and so on. In this cycle, the queries $q_i$ and $q_j$ will be satisfied at time ti and tj accordingly. The cycle finishes at time t3 in the end. For queries $q_i$ and $q_j$, the total access time ($AT$) can be calculated as follows:

$$\begin{aligned} AT &= (ti - t0) + (tj - t1) \\ &= (ti - t2 + t2 - t0) + (tj - t2 + t2 - t1) \\ &= (ti - t2 + tj - t2) + (t2 - t0 + t2 - t1) \end{aligned}$$

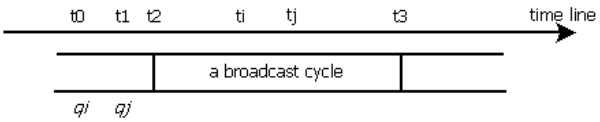



Figure 2: A scenario of on-demand XML data broadcasts

From the above calculation, we can see that the only variable factor that affects AT is the total wait time starting from the beginning the cycle, that is (ti - t2 + tj - t2). As a result, in on-demand XML data broadcasts, the server does not need to consider the arrival time of each query when scheduling XML data on wireless channel but only need to minimize the total query wait time from the beginning of the current broadcast cycle, and in the above case, that is (ti - t2 + tj - t2) for both queries $q_i$ and $q_j$.

### 4.2 LEL (Least Efficient Last) Scheduling Algorithm

LRL scheduling algorithm (Sun et al. 2008) shows the best performance when compared with other existing

multi-item scheduling algorithms. It mainly considers access frequencies of data items and different queried result sets containing multiple items. However, in on-demand XML data broadcasts, the sizes of data items can vary in a very wide range. Thus data item size should be taken into consideration. In this subsection, we devise an improved scheduling algorithm using similar strategy in LRL algorithm.

First of all, we introduce some notations which will be used in the rest of the paper:

- $d_i$: a data item (XML file) stored in the server
- $D$: the set of data items that will be broadcast. $D = d_1, d_2, \ldots, d_m$
- $L_i$: the length of data item $d_i$
- $q_i$: a query issued by one ore more mobile users
- $Q$: the query set $Q = q_1, q_1, \ldots, q_n$ that were issued during the last broadcast cycle
- $QS(d_i)$: the query set in which all queries require data item $d_i$
- $freq(q_i)$: the access frequency of $q_i$
- $\sigma$: the broadcast schedule of a broadcast cycle
- $FQS(d_i)$: given a schedule $\sigma$, the query set that will be fully satisfied by item $d_i$

For a given schedule $\sigma$ and a given query set $Q$, it is easier to identify what queries will be fully satisfied when a data item is broadcast if examining from the last item to the first item in schedule $\sigma$. For example, if $d_m$ is the last item of schedule $\sigma$ and is required by $q_1$, $q_2$ and $q_3$, then all these three queries will only be fully satisfied after they receive $d_m$. Then after removing $d_m$ from $\sigma$ and removing $q_1$, $q_2$ and $q_3$ from $Q$, we perform the same check on the updated $\sigma$ and $Q$. Taking similar steps, we can work out what queries (other than $q_1$, $q_2$ and $q_3$) will be fully satisfied by the new last item of $\sigma$. By doing this repeatedly, the access time of all queries could be easily figured out. Moreover, since the last broadcast item produces the longest access time, it is reasonable that the last item should have the least access frequency. Furthermore, taking the lengths of data items into account, we propose to broadcast the least efficient data item as the last item in a schedule.

**Definition 2.** The efficiency of a data item $d_i$ can be defined as follows:

$$Eff(d_i) = \frac{\sum_{q \in FQS(d_i)} freq(q)}{L_i}$$

Based on this definition, when we schedule a new broadcast cycle, we first examine the efficiency of each data item and then select out the less efficient items to broadcast later and those items with higher efficiency will be broadcast earlier. The LEL (Least Efficient Last) scheduling algorithm can be described in the following.

Note that, in order to calculate $Eff(d_i)$ for the first scheduled data item $d_i$, we initially set all $FQS(d)$ the same as $QS(d)$ for every item $d$ in $D$. Moreover, suppose the data item set $D$ contains $m$ items, then step 1 and step 4 both take $O(m)$ time and step 1 and step 4 will both repeat $m$ times. Therefore, the computing complexity of LEL scheduling algorithm is $O(m^2)$, which is the same as LRL (Sun et al. 2008).

**LEL Scheduling Algorithm:**

1. select an item $d$ from data item set $D$ which has the smallest $Eff(d)$
2. place item $d$ in the last vacant position of broadcast schedule $\sigma$
3. remove item $d$ from $D$
4. update $FQS(d')$ for every item $d'$ in $D$
5. repeat step 1 to step 4 until $D$ becomes empty

LRL algorithm has the property that exchanging the broadcast order of any two successive data items in a schedule $\sigma$ (generated by LRL algorithm) will not achieve better overall access efficiency (Sun et al. 2008). However, LEL algorithm does not guarantee to hold this property.

In order to analyse the similar property of LEL algorithm, according to the finding in subsection 4.1, we only need to consider the wait time starting from the beginning of current broadcast cycle. Suppose two successive data items $d_i$ and $d_{i+1}$ in $\sigma$ (generated by LEL algorithm) will be exchanged, then only the wait time of queries that are fully satisfied by data items $d_i$ and $d_{i+1}$ will be affected. Suppose the total length of data items broadcast before $d_i$ is $L'$, then the total wait time $T$ of those fully satisfied queries (just because of receiving data items $d_i$ and $d_{i+1}$) is

$$\begin{aligned} T &= \sum_{q \in FQS(d_i)} freq(q) \times (L' + L_i) + \\ &\quad \sum_{q \in FQS(d_{i+1})} freq(q) \times (L' + L_i + L_{i+1}) \quad (1) \end{aligned}$$

After data items $d_i$ and $d_{i+1}$ are exchanged, the total wait time $T'$ of those fully satisfied queries for the new schedule $\sigma'$ is

$$\begin{aligned} T' &= \sum_{q \in FQS'(d_{i+1})} freq(q) \times (L' + L_{i+1}) + \\ &\quad \sum_{q \in FQS'(d_i)} freq(q) \times (L' + L_i + L_{i+1}) \end{aligned}$$

Since data items $d_i$ and $d_{i+1}$ are successive, we can infer that after these two items are exchanged, the whole set of fully satisfied queries of these two items remains unchanged. Thus we have

$$FQS(d_i) \cup FQS(d_{i+1}) = FQS'(d_i) \cup FQS'(d_{i+1}) \quad (2)$$

According to the definition of $FQS$, we also have

$$\begin{aligned} FQS(d_i) \cap FQS(d_{i+1}) &= \emptyset \\ FQS'(d_i) \cap FQS'(d_{i+1}) &= \emptyset \end{aligned}$$

Then we have

$$\begin{aligned} T - T' &= \sum_{q \in FQS'(d_{i+1})} freq(q) \times L_i - \\ &\quad \sum_{q \in FQS(d_i)} freq(q) \times L_{i+1} \quad (3) \end{aligned}$$

Therefore, if we have $T - T' \leq 0$, then LEL algorithm holds the similar property of LRL algorithm; if

$T - T' > 0$, we call it a **violation** of the property. In other words, violation means exchanging these two successive data items can result in better access efficiency. However, detecting all these violations in a schedule generated by LEL algorithm could help to improve the overall access efficiency very little. This is because, according to LEL algorithm, when scheduling data item $d_{i+1}$, we must have

$$Eff'(d_{i+1}) \leq Eff(d_{i+1}) \leq Eff'(d_i) \quad (4)$$

Then according to (3) and the definition of $Eff$, if $T - T' > 0$, we must have

$$Eff(d_i) \leq Eff'(d_{i+1})$$

Then we must also have $L_i \leq L_{i+1}$, otherwise if $L_i > L_{i+1}$, we have

$$\begin{aligned}
&\sum_{q \in FQS(d_i) \cup FQS(d_{i+1})} freq(q) \\
&= Eff(d_i) \times L_i + Eff(d_{i+1}) \times L_{i+1} \\
&= Eff(d_i) \times L_{i+1} + Eff(d_{i+1}) \times L_i + [Eff(d_i) \\
&\quad \times (L_i - L_{i+1}) - Eff(d_{i+1}) \times (L_i - L_{i+1})] \\
&< Eff(d_i) \times L_{i+1} + Eff(d_{i+1}) \times L_i \\
&\leq Eff'(d_{i+1}) \times L_{i+1} + Eff'(d_i) \times L_i \\
&= \sum_{q \in FQS'(d_i) \cup FQS'(d_{i+1})} freq(q)
\end{aligned}$$

This is impossible because we have (2). Therefore, we have $L_i \leq L_{i+1}$. Also, if a violation happens, we have $T - T' > 0$. Then according to (3), (4) and the two schedules $\sigma$ and $\sigma'$, we must have

$$\begin{aligned}
T - T' &\leq \sum_{q \in FQS'(d_{i+1})} freq(q) \times L_i \\
&\leq \sum_{q \in FQS(d_{i+1})} freq(q) \times L_{i+1}
\end{aligned}$$

Then according to (1),

$$\begin{aligned}
&\frac{T - T'}{T} \\
&\leq \frac{\sum_{q \in FQS(d_{i+1})} freq(q) \times L_{i+1}}{T} \\
&< \frac{1}{1 + \frac{\sum_{q \in FQS(d_i) \cup FQS(d_{i+1})} freq(q) \times (L' + L_i)}{\sum_{q \in FQS(d_{i+1})} freq(q) \times L_{i+1}}}
\end{aligned}$$

Since for most of i ($1 \leq i < m$) in schedule $\sigma$, we have

$$L' + L_i = L_1 + L_2 + \ldots + L_i \gg L_{i+1}$$

Therefore we have

$$\begin{aligned}
&\sum_{q \in FQS(d_i) \cup FQS(d_{i+1})} freq(q) \times (L' + L_i) \\
&\gg \sum_{q \in FQS(d_{i+1})} freq(q) \times L_{i+1}
\end{aligned}$$

Then we can infer that if $T - T' > 0$, or in other words, if a violation happens, in most cases we should have

$$\frac{T - T'}{T} \to 0$$

As a result, most violations (if exist) cannot help to effectively improve the access efficiency. We will present the experimental results of the analysis in the next section.

## 5 Experiments

In this section, we mainly study the performance of *combining scheme* described in Section 3 and the effectiveness of LEL scheduling algorithm in previous section.

In our experiments, synthetic XPath queries are generated using the generator developed by Diao et al. (2003). All queries are distinct. The maximum depth of XPath queries is 6. Experiments are run on a synthetic data set: News Industry Text Format (NITF) DTD, and 500 XML documents are generated. The average depth of all documents is about 6.

Table 2 shows the descriptions of three parameters are varied in the experiments: the threshold of *sim* ($T_{sim}$), the number of queries ($N_q$), and the probability of * and // in each query's step (*prob*).

Table 2: Workload parameters for our experiments

| Parameter | Range | Default Value |
|---|---|---|
| $T_{sim}$ | 1 to 16 | 1.4 |
| $N_q$ | 100 to 500 | 300 |
| *prob* | 10% to 50% | 30% |

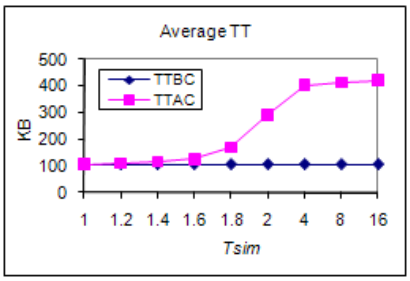
### 5.1 Performance of Combining Scheme

Figure 3 (a) presents the average tuning time of each query when varying $T_{sim}$. TTBC denotes the average tuning time before using *combining scheme* while TTAC denotes the average tuning time after. When $T_{sim}$ is 1, the average tuning time is minimized and is equal to that of *snippet scheme*. When $T_{sim}$ is small, the amount of redundant data incurred by *combining scheme* keeps small as well. Also from the figure we can see that when $T_{sim}$ is smaller or equal to 1.6, the average tuning time increases very slowly. After that it grows rapidly. This is because a larger $T_{sim}$ can result in more redundant data to download for each query. However, when $T_{sim}$ is greater than 4, the average tuning time becomes stable since the redundant data is very close to that of *atomic scheme*, which contains the most redundant data amount all three schemes.

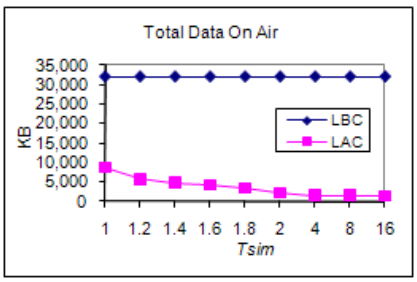
Figure 3 (b) shows the total data on the wireless channel when varying $T_{sim}$. LBC denotes the total length of data on the wireless channel before using *combining scheme* while LAC denotes the total length of data on the wireless channel after. As shown in the figure, the total length of data drops sharply by using our *combining scheme*. When $T_{sim}$ is 1, the length is almost one third of original data using *snippet scheme*. This is because although each query is different, but the matched snippets or sub-trees of each XML file are still likely the same. When $T_{sim}$ continues to grow, the total length of data continues to drop. However, when $T_{sim}$ is greater than 4, the total length of data becomes stable since the combined results are very close to that of *atomic scheme* and thus most possible combinations of snippets have been performed.

We study the effectiveness of LEL scheduling algorithm by comparing it with LRL scheduling algorithm (Sun et al. 2008), since it show the best access efficiency for multi-item on-demand data broadcasts in

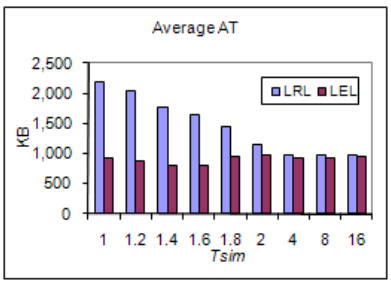
previous work. Figure 3 (c) demonstrates the comparison[2] between LEL and LRL scheduling algorithms when varying $T_{sim}$. From the figure, we can see that LEL is always better than LRL. When $T_{sim}$ is very large, the gap between LEL and LRL becomes small since each query tends to require much more data due to combined and need to download much more redundant data. When $T_{sim}$ is 1.4, the LEL achieves the best access efficiency. Meanwhile, from Figure 3 (a), we can see that the overhead of tuning time is quite low when $T_{sim}$ is 1.4. Therefore, in *combining scheme*, by splitting query results into XML snippets and combining similar ones, the average access time can be effectively reduced (by about 14%) only with a little overhead (about 8%) to the optimal tuning time. In addition, when $T_{sim}$ is 1.4, if compared with *atomic scheme* which is adopted in previous work, the tuning time can be reduced by 75%.




(a)




(b)




(c)

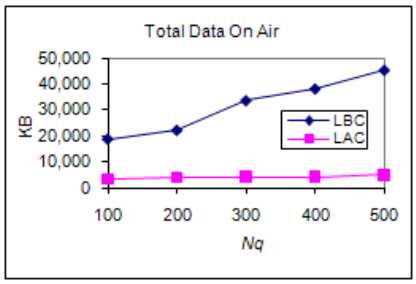
Figure 3: Varying $T_{sim}$

[2]Note that, according to subsection 4.1, we only compare the average wait time starting from the beginning of the current broadcast cycle (we still denote it as average AT in the figure, since it will not affect our comparisons).

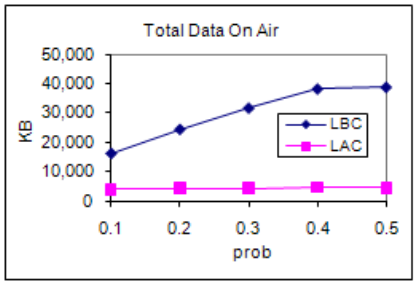
### 5.2 Performance of LEL Algorithm

Figure 4 (a) presents the comparison between LEL and LRL scheduling algorithms when varying $N_q$. Generally, the average access time of LEL algorithm is only about 50% of LRL algorithm. When $N_q$ increases, the average access time of both LEL and LRL algorithms increases slowly as well because the more queries were issued to the server, the more data would be required and thus could result in more access time.

Figure 4 (b) depicts the comparison between LEL and LRL scheduling algorithms when varying *prob*. The average access time of LEL algorithm is about 40% to 65% of LRL algorithm. When prob increases, the average access time of both LEL and LRL algorithms increases as well. The reason for this is similar to Figure 4 (a). When prob is larger, more data will be required and thus could result in more access time. Moreover, when prob is 10%, the average access time of LEL algorithm is about 40% of LRL algorithm and when prob is 50%, the average access time of LEL algorithm grows faster than LRL algorithm and is about 65% of it.




(a) Varying $N_q$




(b) Varying *prob*

Figure 4: Performance of LEL algorithm

Table 3 shows the results of fixing violations[3] (defined in subsection 4.2). From the table, we can see that the average number of violations is about 875. However, the average improvement is only about 0.3%. Therefore, detecting and fixing all violations in the schedule generated by LEL algorithm brings very limited improvements to the overall access efficiency. Also, the results in the below table match our analysis in subsection 4.2 very well.

[3]LEL-VA algorithm examines any two successive data items in the schedule generated by LEL algorithm, like $d_i$ and $d_{i+1}$, to check whether a violation happens. If so, it exchanges the order of these two data items. The algorithm continues to run until all violations have been detected and fixed.

Table 3: Violations fixing results

| $N_q$ | Average AT (LEL) - KB | Average AT (LEL-VA) - KB | Number of violations | Improvement (%) |
|---|---|---|---|---|
| 100 | 673.11 | 672.22 | 249 | 0.13 |
| 200 | 726.7 | 724.27 | 736 | 0.33 |
| 300 | 770.7 | 765.96 | 1596 | 0.62 |
| 400 | 842.38 | 841.61 | 491 | 0.09 |
| 500 | 847.07 | 844.19 | 1305 | 0.34 |

## 6 Conclusions

In this paper, we propose to pre-process XML data before broadcast to enhance the overall performance of on-demand XML data broadcasts. Firstly, taking advantage of the structured characteristics of XML data, we discuss two naive schemes (*atomic scheme* and *snippet scheme*) on how to match XML data with mobile users' queries and based on these two schemes, a more practical scheme named *combining scheme* is put forward. It further makes use of the similarity of different XML snippets. Moreover, we analyze the scheduling problem of on-demand XML data broadcasts and define the efficiency of a data item. Based on this definition, we then devise a Least Efficient Last (LEL) scheduling algorithm to effectively organize XML data on wireless channels. Computing complexity and theoretical analysis of LEL scheduling algorithm are also given.

In our experiments, in *combining scheme*, by splitting query results into XML snippets and combining similar ones, the average access time can be effectively reduced (by about 14%) with a little overhead (about 8%) to the optimal tuning time in theory. More importantly, if compared with *atomic scheme* which is commonly adopted in previous work, the tuning time can be reduced by up to 75%. Hence, *combining scheme* can provide much better balance between access efficiency and power efficiency. Furthermore, in our experiments, the average access time of LEL algorithm is only about 40% to 65% of LRL algorithm. Therefore, by using LEL scheduling algorithm, the access efficiency can be improved significantly. In addition, we also demonstrate both theoretically and experimentally that if we detect and fix all violations of any two successive data items in a schedule $\sigma$ generated by LEL algorithm, the average improvement of the overall access efficiency is only about 0.3%, which is quite limited.

## References


3GPP2 (2007), *3rd Generation Partnership Project 2, http://www.3gpp2.org*.

Acharya, S. & Muthukrishnan, S. (1998), Scheduling on-demand broadcasts: New metrics and algorithms, *in* 'MOBICOM', pp. 43–54.

Acharya, S., Alonso, R., Franklin, M. J. & Zdonik, S. B. (1995), Broadcast disks: Data management for asymmetric communications environments, *in* 'SIGMOD Conference', pp. 199–210.

Aksoy, D. & Franklin, M. J. (1999), 'RxW: a scheduling approach for large-scale on-demand data broadcast', *IEEE/ACM Trans. Netw.* **7**(6), 846–860.

Chang, Y.-I. & Hsieh, W.-H. (2004), An efficient scheduling method for query-set-based broadcasting in mobile environments, *in* 'ICDCS Workshops', pp. 478–483.

Chung, Y. D. & Kim, M.-H. (1999), QEM: A scheduling method for wireless broadcast data, *in* 'DASFAA', pp. 135–142.

Chung, Y. D. & Lee, J. Y. (2007), 'An indexing method for wireless broadcast xml data', *Inf. Sci.* **177**(9), 1931–1953.

Diao, Y., Altinel, M., Franklin, M. J., Zhang, H. & Fischer, P. M. (2003), 'Path sharing and predicate evaluation for high-performance xml filtering', *ACM Trans. Database Syst.* **28**(4), 467–516.

Huang, Y., Zhang, Y. & He, J. (2010 (in print)), 'Optimizing broadcast completion time by rescheduling worst-bandwidths-links first in clouds environments', *Journal of Computer and System Science*.

Imielinski, T., Viswanathan, S. & Badrinath, B. R. (1997), 'Data on air: Organization and access', *IEEE Trans. Knowl. Data Eng.* **9**(3), 353–372.

Lee, G., Yeh, M.-S., Lo, S.-C. & Chen, A. L. P. (2002), A strategy for efficient access of multiple data items in mobile environments, *in* 'Mobile Data Management', pp. 71–78.

Lee, W.-C. & Zheng, B. (2005), DSI: A fully distributed spatial index for wireless data broadcast, *in* 'ICDE', pp. 417–418.

Park, C.-S., Kim, C. S. & Chung, Y. D. (2005), Efficient stream organization for wireless broadcasting of xml data, *in* 'ASIAN', pp. 223–235.

Park, S.-H., Choi, J.-H. & Lee, S. (2006), An effective, efficient xml data broadcasting method in a mobile wireless network, *in* 'DEXA', pp. 358–367.

Qin, Y., Sun, W., Zhang, Z., Yu, P., He, Z. & Chen, W. (2009), A novel air index scheme for twig queries in on-demand xml data broadcast, *in* 'DEXA', pp. 412–426.

Sun, W., Shi, W., Shi, B. & Yu, Y. (2003), 'A cost-efficient scheduling algorithm of on-demand broadcasts', *Wireless Networks* **9**(3), 239–247.

Sun, W., Yu, P., Qin, Y., Zhang, Z. & Zheng, B. (2009), Two-tier air indexing for on-demand xml data broadcast, *in* 'ICDCS', pp. 199–206.

Sun, W., Zhang, Z., Yu, P. & Qin, Y. (2008), Efficient data scheduling for multi-item queries in on-demand broadcast, *in* 'EUC (1)', pp. 499–505.

Vagena, Z., Moro, M. M. & Tsotras, V. J. (2007), RoXSum: Leveraging data aggregation and batch processing for xml routing, *in* 'ICDE', pp. 1466–1470.

Xu, J., Lee, D.-L., Hu, Q. & Lee, W.-C. (2002), *Handbook of wireless networks and mobile computing*, John Wiley & Sons, Inc., pp. 243–265.