PrefixSummary: A Directory Structure for Selective Probing on Wireless Stream of Heterogeneous XML Data

Chang-Sup PARK†, Jun Pyo PARK††, Nonmembers, and Yon Dohn CHUNG††(a), Member

SUMMARY Wireless broadcasting of heterogeneous XML data has become popular in many applications, where energy-efficient processing of user queries at the mobile client is a critical issue. This paper proposes a new index structure for wireless stream of heterogeneous XML data to enhance tuning time performance in processing path queries on the stream. The index called PrefixSummary stores for each location path in the XML data the address of a bucket in the stream which contains an XML node satisfying the location path and appearing first in the stream. We present algorithms to generate broadcast stream with the proposed index and to process a path query on the stream efficiently by exploiting the index. We also suggest a replication scheme of PrefixSummary within a broadcast cycle to reduce latency in query processing. By analysis and experiment we show the proposed PrefixSummary approach can reduce tuning time for processing path queries significantly while it can also achieve reasonable access time performance by means of replication of the index over the broadcast stream.

key words: wireless broadcast, XML data stream, index, XML path query processing

1. Introduction

Recently, wireless information services have become popular in industry with the development of wireless communication technologies such as Mobile WiMAX [13] and LTE [15]. Mobile users carry small hand-held devices such as smart phones and PDAs with limited data processing capabilities. In the wireless environment, broadcasting is an effective means for providing information to a large number of mobile clients since it has the benefits of bandwidth efficiency, scalability, and energy-efficiency [1], [6], [8]. The information server disseminates data through a broadcast channel, on which mobile clients listen and download data of their interests without sending any requests to the server.

With the successful development and proliferation of related technologies and applications, XML (eXtensible Markup Language) [4] has been used as a standard language to facilitate the representation and sharing of semi-structured data across different information systems not only in the wired Internet but also in the wireless broadcast environment. Currently, various XML data are disseminated over the wireless network in practical applications such as Electric Program Guide (EPG) of Digital Audio/Video Broadcasting services [7] and Traffic and Travel Information (TTI) in TPEG services [10]. In such applications, XML data often have heterogeneous schemas. For example, the XML data of news articles, stock reports, traffic updates, and weather forecasts are based on different schemas and are often provided from different sources (i.e. information providers).

In the wireless data broadcasting, both latency-efficiency and energy-efficiency in accessing data on the stream should be considered. Latency-efficiency means that the overall processing time of a query should be minimized to provide fast response to the user. Meanwhile, energy-efficiency means that since the mobile device is battery-powered, selective access of the broadcast stream is necessary for energy conservation in the device. Those performance aspects are usually measured by access time and tuning time, respectively. The former is the time elapsed from the time a query is issued at the mobile device to the time the target data of the query is completely retrieved from the stream. The latter is the sum of the elapsed times when the mobile device tunes in and receives data from the broadcast stream in the active mode, in which the device consumes much more energy than when it is in the doze mode [9].

In this paper, we study wireless broadcasting of heterogeneous XML documents which enables energy efficient processing of user queries at the mobile client. Heterogeneous XML data have their own root elements of different names, and thus absolute location paths belong to the different schemas can be distinguished by the root element’s name. We consider location path queries on the XML data, which are commonly used in the query languages for XML such as XPath [2] and XQuery [3]. A location path query consists of a sequence of multiple location steps, each of which selects a set of elements or text nodes in the document tree. To evaluate location path queries on the broadcast stream of heterogeneous XML data efficiently, we propose a new index structure called PrefixSummary which stores for each location path in the XML data the address of the first XML node on the stream that satisfies the location path. Our approach exploits existing XML streaming methods such as S-Node [11] and DIX [12] to generate the stream of each XML document and process user queries by using special link indices supported by them, but PrefixSummary can be used with any streaming method such as the naive text serialization. We also suggest replication of the PrefixSummary index in a broadcast stream to reduce latency in query processing. We provide algorithms to generate broad-
cast stream with PrefixSummary and to process a location path query on the stream by exploiting the proposed index. We show by analysis and experiment that our approach can achieve significant improvement on tuning time in processing user queries.

The rest of the paper is organized as follows. We first discuss related work and motivation in Sect. 2. In Sect. 3, we propose a broadcast stream structure for XML data including PrefixSummary and describe stream generation and query processing algorithms. We analyze the performance of the proposed approach in Sect. 4 and show the result of experimental evaluation in comparison with the conventional methods in Sect. 5. Finally, we draw a conclusion in Sect. 6.

2. Related Work and Motivation

In the literature of wireless XML broadcasting, there have been proposed several approaches for energy-efficient query processing over the XML stream. Park et al. [11] proposed a broadcast stream organization for XML data called S-Node. It broadcasts XML data as stream of S-nodes which represent elements in XML data. Each S-node contains three kinds of links to the other S-nodes. A same-tag sibling link points to the nearest sibling S-node that represents an element with the same tag name. A different-tag sibling link points to the nearest sibling S-node representing an element with a different tag name. Last, if an S-node is the last one among its sibling nodes (i.e. the last child node of its parent node), it may have a same-path link to the nearest S-node representing an element with the same location path specification. By exploiting these links in S-nodes, the mobile client can avoid tuning of irrelevant data and find answer nodes satisfying a given user query in an efficient way.

Park et al. [12] proposed distributed index structures for streaming XML data, called DIX and c-DIX. In those approaches, XML data are broadcast in the stream of DIX nodes. A DIX node represents an XML element and contains the location path specification of the element it represents. Thus, the mobile client can start evaluation of a user query from the tune-in point on the stream without waiting for a document root element in the next broadcast cycle. In addition, it can skip irrelevant nodes by using a Clone-node Link and Foreign-node Link contained in each DIX node. A clone-node link in a DIX node stores the address of the nearest node which has the same depth and location path. In processing a user query, the clone-node link in an answer node can be used to find the next answer node directly from the stream. A foreign-node link in a DIX node stores the address of the nearest node which has the same depth but has a different location path. Thus, it can be used to skip irrelevant nodes, which have different location paths from that of the query, efficiently. In the c-DIX scheme, DIX nodes are clustered in the stream based on the length of location paths to further improve access time and tuning time.

These approaches commonly considered broadcasting of a single XML document. If a set of heterogeneous XML data of different schemas are disseminated over a wireless channel, when a user query is given for the XML data at the mobile client, the conventional methods should first find a target document containing the answers to the given query by probing the entire stream of XML data sequentially.

In the previous stream organization methods, once an answer node to the given query is found in the stream, the other answer nodes following the first one can be accessed efficiently by exploiting unique link or index information provided in the stream structures (e.g., the same-tag sibling links and same-path links in the S-Node stream and the Clone-node Links in the DIX stream). However, those methods should commonly explore XML data in the stream until they find the first answer node. This often requires a significant amount of tuning time in processing user queries.

To solve this problem, we propose a new index structure called PrefixSummary to store the address of a bucket containing the first answer node for each path specification of the XML data. By exploiting the pre-calculated addresses, we can find and retrieve the first answer node to the query directly from the stream, without performing sequential probing of different kinds of XML data to find the target document and exploration of the target document to find the first answer of the query.

3. PrefixSummary: Our Proposed Method

3.1 Broadcast Stream Structure

We consider that a broadcast stream is built from a set of XML documents having different schemas. We first define a new index, called PrefixSummary, for the stream of heterogeneous XML documents.

Definition 1. PrefixSummary of depth \( h \), denoted by \( PS(h) \), is a set of index entries \((p, a)\) where \( p \) is a location path of length \( h \) extracted from the given XML documents and \( a \) is the address of a bucket containing the XML node which appears first among all the answer nodes satisfying \( p \) in the stream. \( \square \)

In the above definition, the length of a location path is the number of location steps contained in the path. We assume that location paths considered for indexing are specified from the root element of an XML document. The address of a bucket means the time of delivery of the bucket on the wireless broadcast channel.

In our approach, a broadcast stream consists of two parts: PrefixSummary index and the content of the given XML documents. PrefixSummary is attached before the XML data stream and it can be also replicated within the XML data in a broadcast cycle, as will be described in Sect. 3.3. Figure 1 shows an example broadcast stream having a PrefixSummary index of depth 2. The index contains link addresses for different location paths of length 2 derived from the XML data stream. For instance, it has an entry for a location path ‘/s/z’ which stores the address \( t_0 \) of a bucket containing an element that satisfies the location path and ap-
pears first in the XML data stream.

We suppose that a specific streaming method is used to construct data stream for each XML document. It can be the straightforward text serialization of XML documents or one of the existing XML streaming methods based on unique distributed indices, such as S-Node [11] or DIX [12]. Note that our approach is independent of the stream organization and query processing scheme employed in the XML broadcast system.

Figure 2 shows an algorithm to generate the proposed broadcast stream. For each XML document, it first generates a broadcast stream using a specific streaming method \( \sigma \) and then concatenates it to the previously generated stream of XML documents (line 3~4). As shown in line 6~12, the generated XML data stream is exhaustively examined to identify the positions of XML elements having a different location path and to generate index entries of PrefixSummary in the form of \((\text{a location path specification, a bucket address of the first answer node})\), as defined in Definition 1. Finally, the PrefixSummary index is constructed based on the index entries extracted from the entire XML data stream and is attached before the XML data stream (line 15~16).

![Fig. 1](image_url)

**Algorithm 1 Broadcast Stream Generation**

**Input:** a set \( D \) of XML documents, path length \( h \)

**Output:** XML broadcast stream \( S \)

1. \( \text{IndexEntries} := \emptyset; \text{DataStream} := \emptyset; \)
2. FOR-EACH (XML document \( d \) in \( D \)) {
3. Generate broadcast stream \( S_d \) of \( d \) using a streaming method \( \sigma \);
4. \( \text{DataStream} := \text{DataStream} \cup S_d \);
5. \( \text{Paths} := \emptyset; \)
6. FOR-EACH (element \( E \) in \( S_d \) in the order of appearance in \( S_d \)) {
7. \( p := \text{prefix of length } h \) of the location path specification of \( E \);
8. IF \((p \text{ is not in } \text{Paths})\) {
9. \( a := \text{address of the bucket in } \text{DataStream} \text{ which contains } E; \)
10. Add a pair \((p, a)\) to \( \text{IndexEntries}; \)
11. Add a prefix \( p \) to the \( \text{Paths}; \)
12. }
13. }
14. }
15. Construct \( PS(h) \) from \( \text{IndexEntries}; \)
16. Place \( PS(h) \) and \( \text{DataStream} \) into a broadcast stream \( S \);
17. RETURN \( S; \)

![Fig. 2](image_url)

3.2 Query Processing over Broadcast Stream

Figure 3 shows a generalized algorithm to access broadcast stream with PrefixSummary and evaluate a given query. It first finds a PrefixSummary index nearest from the tune-in point on the stream (line 2~5). It is assumed that each bucket \( B \) contains a type specifier \( \text{TYPE} \) as well as an address \( \text{PSLink} \) of the next nearest PrefixSummary index, and thus, the mobile client can locate the first bucket of a PrefixSummary index from any tune-in point on the stream. Then the algorithm searches the index for an entry having a location path that matches a prefix of the location path specification \( p \) of the given query (line 6). If such entry is found, the target address in the entry is exploited to access a bucket directly from the stream, which contains a node satisfying the prefix of \( p \). The operation \( \text{getFirstAnswer()} \) in line 7 finds the first answer to the location path \( p \) contained in the relevant XML document. Subsequently, the algorithm explores data stream of the selected XML document to find all the other answer nodes for \( p \) (line 9~14). Note that the operation \( \text{getNextAnswer()} \) in line 11 finds the next answer node appearing in the stream efficiently by using access methods supported by the adopted XML stream structure. In the S-Node stream [11], for example, same-tag sibling links and same-path links in S-nodes, which have the address of the nearest node with the same path specification, can be used to access only the answer nodes to the query in the stream.

![Fig. 3](image_url)

3.3 Replication of PrefixSummary

In the wireless data dissemination, the notion of index replication is known to improve the latency efficiency of mobile clients, since they can find the target data in a shorter latency when index buckets are frequently placed over the broadcast stream. Among a variety of index replication approaches,
we adopt the (1, M) index replication policy [5] for our PrefixSummary replication due to its simplicity and efficiency.

In the (1, M) index replication method, index information is replicated and placed M times in a single broadcast cycle (i.e. bcast). That is, the index is placed equidistantly before every 1/M fraction of the data. For this, Algorithm 1 (Broadcast Stream Generation) in Fig. 2 needs to be modified, but Algorithm 2 (Query Processing) in Fig. 3 does not. Figure 4 shows an example of PrefixSummary PS(2) replicated in a broadcast cycle. In the replicated PrefixSummary, denoted as PS₂ in the figure, the address values of the index entries for location paths '/a/b,' '/a/c,' '/r/a,' '/r/s,' '/s/b,' and '/s/z' should be changed compared with the original PrefixSummary PS₁, since the target buckets of those path specifications are delivered in the next broadcast cycle. Note that as shown in Fig. 4, if the tune-in point of a user query is on the stream of d₁ or d₂ and the target document of the query is d₃, PS₂, instead of PS₁ at the beginning of the next broadcast cycle, can be used to address the first answer node in d₃ which is broadcast in the same cycle. Therefore, the access time of data stream can be reduced by exploiting replicated indices. For simplicity of the index generation and query processing, we assume that if a replica of PrefixSummary is placed within the stream of a particular XML document, like PS₂ within d₃ in Fig. 4, it stores addresses for the document broadcast in the next cycle.

4. Analysis

In this section, we analyze the latency efficiency and energy efficiency of the proposed scheme in terms of access time and tuning time of the broadcast stream. In the rest of the paper, for convenience but without loss of generality, the access and tuning time as well as the size of XML data stream are measured in the unit of buckets since bandwidth of a wireless channel is often fixed in the real environment.

Table 1 shows notation of the parameters used for the performance analysis in the paper. Note that ∆data is not the size of original XML documents in the disk but the size of XML data placed on the broadcast stream via a specific XML streaming method such as S-Node.

We suppose that XML documents are serialized into a broadcast stream by a pre-defined XML streaming method σ. We also assume that a user query is given for only one of the XML documents, called a target document, which contains all the answers to the query. Average access time and tuning time to evaluate a given query on its target XML document depend on the employed streaming method σ. They are denoted by α and τ, respectively, in this paper.

4.1 Broadcast Stream without PrefixSummary

Figure 5-(a) depicts a query evaluation process on the stream of heterogeneous XML documents generated by a conventional streaming method without PrefixSummary. For a mobile client to access the target data on the broadcast stream, it firstly has to find the starting bucket of an XML document closest to the first tune-in point on the stream (i.e. probe wait). Since the interval between starting buckets of two adjacent XML documents is ∆data/d, the average probe wait is 0.5∆data/d. After reaching the start bucket (i.e., containing the root node) of an XML document, the mobile client should find the relevant XML document which can answer the given user query by investigating only the start buckets of the subsequent XML documents delivered on the air. That is, a half of the start buckets of the XML documents, on the average, should be examined to find the target document of the query, assuming that user queries are evenly distributed over all documents. Thus, the access time required for this (called, the target document wait) is 0.5(1-1/σ)∆data. Finally, the mobile client evaluates the query on the selected XML document stream in target document access time α. In consequence, the average access time is computed as fol-
\[ \text{AvgAT}_\sigma = 0.5 \frac{\Delta \text{data}}{d} + \frac{0.5}{d} \left( 1 - \frac{1}{d} \right) \Delta \text{data} + \alpha \\
= 0.5 \Delta \text{data} + \alpha \]  
(1)

For the probe wait period, one bucket (i.e., the initial probing bucket) must be tuned. If this bucket happens to be the start bucket of an XML document, this bucket tuning becomes a part of the following target document wait period. In the rest of the paper, we ignore this special case in the analysis. Then, a half of the start buckets of the documents, on the average, are examined in the target document wait period to find the target document, and the query is processed over the document in target document tuning time \( \tau \). Therefore, the average tuning time is

\[ \text{AvgTT}_\sigma = 1 + 0.5d + \tau \]  
(2)

4.2 Broadcast Stream using PrefixSummary

Figure 5-(b) shows a query evaluation process on the broadcast stream with PrefixSummary. In this case, the access time of a user query consists of four parts: (a) probe wait time, (b) PrefixSummary probe time, (c) target document wait time, and (d) target document access time. The first is the time required to reach the nearest PrefixSummary from the position the user initially tuned in. Since PrefixSummary is placed \( m \) times within a bcast, the average probe wait is computed as \( 0.5 \Delta \text{bcast}/m = 0.5(\Delta \text{data}/m + \Delta \text{PS}(h)) \). PrefixSummary probe time equals to the size of PrefixSummary in buckets, \( \Delta \text{PS}(h) \). The target document wait is the time required for the target document of the query to arrive on the air after a replica of PrefixSummary is read. Assuming that target documents of user queries are uniformly distributed over all the documents, it is computed as the average interval from the end of the probed PrefixSummary index to the beginning of a document in the stream, i.e.,

\[ 0.5 \cdot \frac{\Delta \text{bcast}}{d} + \frac{1}{d} \sum_{i=0}^{d-1} \frac{\Delta \text{bcast}}{d} - i \]

Finally, the target document access time is the elapsed time in accessing the target XML document to find query results. As mentioned earlier, it is considered as a constant time which is dependent on the adopted streaming method. Therefore, the average access time of the stream based on PrefixSummary is computed as follows:

\[ \text{AvgAT}_{PS+\sigma} = 0.5 \left( \frac{\Delta \text{data}}{m} + \Delta \text{PS}(h) + \Delta \text{PS}(h) \right) \\
+ 0.5 \left( \frac{d^2 - 2d + 2}{2d^2} \right) \left( \Delta \text{data} + m \Delta \text{PS}(h) + \alpha \right) \\
= 0.5 \left( \frac{1}{m} + \frac{d^2 - 2d + d}{d^2} \right) \Delta \text{data} \\
+ 0.5 \left( \frac{d^2 - 2d + 2}{d^2} m + 3 \right) \Delta \text{PS}(h) + \alpha \]  
(3)

By subtracting Eq. (1) from (3), we have

\[ \text{AvgAT}_{PS+\sigma} = \text{AvgAT}_{\sigma} + 0.5 \left( \frac{1}{m} - \frac{2(d - 1)}{d^2} \right) \Delta \text{data} \\
+ 0.5 \left( \frac{d^2 - 2d + 2}{d^2} m + 3 \right) \Delta \text{PS}(h) \]  
(4)

In case of the broadcast stream including PrefixSummary, its tuning time in the unit of buckets is easily computed as the sum of (a) one bucket of initial tuning, (b) the buckets read in probing PrefixSummary, and (c) a sequence of bucket tunings within a single target XML document. The last one, called target document tuning, is performed based on the specific streaming method employed. However, it can be done in much shorter time on the stream with PrefixSummary than on the conventional stream without PrefixSummary since the bucket containing the first answer to the query can be directly identified and accessed by using its address obtained from PrefixSummary. That is, tuning time \( \tau_f \) to search for the first answer node in the target document can be saved in our query processing scheme. Therefore, the average tuning time in the proposed stream can be computed as follows:

\[ \text{AvgTT}_{PS+\sigma} = \text{AvgTT}_{\sigma} + 0.5 \Delta \text{PS}(h) - (0.5d + \tau_f) \]  
(5)

5. Performance Evaluation

5.1 Experiment Settings

To show effectiveness and performance of the proposed approach, we have conducted experiments using real data sets and present their results in this section. We have used three XML documents with different schemas, Mondial, University Courses from Reed College, and SIGMOD Record, as test data sets, which were selected from the XML data repository by the University of Washington’s Database Research Group [14]. The characteristics of the XML documents are as follows:

- Mondial has 22,423 elements and the maximum depth of the elements is 5.
- University courses from Reed College has 10,546 elements and the maximum depth of the elements is 4.
- SIGMOD Record has 11,526 elements and the maximum depth of the elements is 6.

In the experiments, we have generated heterogeneous broadcast streams containing the above XML documents in the order by using 6 XML streaming methods: text serialization, three methods (i.e., SL, SD, and SP) from the S-Node scheme [11], and two methods (i.e., DIX and c-DIX) from the DIX scheme [12]. The size of the XML data streams (in buckets) generated by the different streaming methods are presented in Table 2, where the bucket size is 256 Bytes.
Table 2  Size of XML data streams (in buckets).

| Streaming Method | TS   | SL    | SD    | SP    | DIX  | c-DIX |
|------------------|------|-------|-------|-------|------|-------|
| Mondial          | 4,193| 5,115 | 5,468 | 5,468 | 5,115| 6,423 |
| Reed             | 792  | 1,165 | 1,335 | 1,335 | 1,165| 1,740 |
| SigmodRecord     | 1,434| 1,824 | 2,008 | 2,008 | 1,824| 2,713 |
| Total            | 6,419| 8,104 | 8,811 | 8,811 | 8,104|10,876 |

Table 3  Size of PrefixSummary (in buckets).

| Depth of PrefixSummary | 1 | 2 | 3 | 4 | 5 | 6 |
|------------------------|---|---|---|---|---|---|
| Size of PrefixSummary | 1 | 2 | 5 | 7 | 8 | 9 |

Table 4  Test queries.

| Query | XPath Specification | Target Document |
|-------|---------------------|-----------------|
| M1    | /mondial/island     | Mondial         |
| M2    | /mondial/country/region/history/ancestor | | |
| M3    | /mondial/country/name[@name='Tyrol'] | | |
| R1    | /root/course/time   | Reed            |
| R2    | /root/course/place/building | | |
| R3    | /course             | | |
| S1    | /SigmodRecord/issue/number | Sigmod Record | |
| S2    | /SigmodRecord/issue/articles/article | | |
| S3    | /SigmodRecord/issue/*/article   | | |

We constructed broadcast streams of the XML data having PrefixSummary and evaluated the performance of query processing on the streams in comparison with the streams without PrefixSummary. We also conducted experiments by varying the depth of PrefixSummary and the number of its replicas placed in the stream. The size of the PrefixSummary indices of different depths is shown in Table 3. As shown in the Table 2 and Table 3, the size of the PrefixSummary indices are significantly small compared with the generated XML data streams. Note that PrefixSummary is independent of a streaming method used to generate data stream.

It should be also noted that since the above XML documents have regular structures with respect to the occurrences of different types of elements in the documents, the first answer to a location path query usually appears early in the target data stream. To show the effectiveness of the proposed approach more clearly, we have inserted some dummy elements in the XML documents. Specifically, 100 elements are added per level in the document trees hence their total amount is about 4% of the number of the original elements.

To evaluate the performance of query processing on the broadcast streams, we used various XPath queries shown in Table 4.

5.2 Experiment Results

In the first experiment, the queries in Table 4 have been evaluated on the test XML streams using the query processing algorithm presented in Sect. 3.2. Each query was evaluated 10 times using different tune-in points which are uniformly distributed over a single broadcast cycle of stream. In the experiment, the depth of PrefixSummary is 6 and the number of replicas of the index in a broadcast cycle is 1. Figure 6 and Fig. 7 respectively show the average tuning time and access time (in the unit of buckets) in processing the test queries on the considered XML streams generated by different streaming methods.

As shown in Fig. 6, tuning time performance of the heterogeneous XML stream improves by using the proposed PrefixSummary index. Especially, tuning time of query M1 on the Mondial data was reduced by 69.1% at the SL method based stream and 98.8% at the DIX method based stream, compared with the query processing results over the streams without PrefixSummary. Experiments with query M2 on the same data set have achieved tuning time improvement ranging from 60.2% at the text serialization stream to 98.6% at the SP stream. A significant improvement of tuning time achieved for queries M1 and M2 in most of the streams is due to the fact that the first elements satisfying the queries appear at the latter part of the Mondial data stream and Mondial has a large number of child elements at the second and third level in its document tree. In addition, tuning times for query S3 (i.e., descendant axis) were significantly reduced at the considered streams. By exploring index entries in PrefixSummary, we can infer that the answer nodes satisfying the axis are elements of path `/root/course.' Thus,
we can further improve the tuning time by omitting unnecessary downloads. For the other test queries, tuning time was shown to be reduced by less than about 10%. By using PrefixSummary, we can avoid large amount of probing into the target data stream which should be performed before the first answer to the query is found in the S-Node and DIX based streams. Also, we can select the candidate nodes for the query involving descendant axis.

On the other hand, access time performance of the XML stream having PrefixSummary has been aggravated compared with the stream without PrefixSummary, as shown in Fig. 7. This is because as shown in Fig. 5, the proposed scheme should first find and probe the nearest PrefixSummary in the stream while query processing on the naive stream without PrefixSummary begins with sequential search of XML documents from the tune-in point in the stream. If only one instance of PrefixSummary is placed in a broadcast cycle as in this experiment, the mobile client has to wait until the PrefixSummary index is delivered at the beginning of the next broadcast cycle. This problem can be partly solved by replicating PrefixSummary within a single broadcast cycle of stream, which will be shown in the result of the later experiment.

In the second experiment, we evaluate the performance of the PrefixSummary index with respect to its depth $h$, i.e., the length of the location paths stored in the index. We have executed test queries on the broadcast streams having PrefixSummary of depth 1 to 6. Average tuning time and access time results on the stream generated by SL method are presented in Fig. 8 and Fig. 9, respectively. The other streaming methods have shown the similar results.

Figure 8 shows that as the depth of PrefixSummary increases, tuning time in processing a path query reduces until the depth becomes equal to the length of the given location path. This is because when a location path in the index matches a prefix of the query, the longer (i.e., more specific) location path helps to get closer to the first answer node of the query in the stream. However, when the location paths in the index become longer than the query, overall tuning time rather increases since the location paths more specific than the query bring no more benefit to finding the answer of the query while they increase the size of PrefixSummary to be probed.

We observe that query M2 shows significant reduction in its tuning time when the depth of PrefixSummary increases from 2 to 3. This is due to the fact that Mondial document has a large number of elements satisfying `/mondial/country`, which also have many child elements other than `region`, while there are few elements satisfying `/mondial/country/region` which appear late in the broadcast stream. Therefore, the index entry pointing to the first occurrence of the `/mondial/country/region` element in the stream provides a large benefit to the overall tuning time for query M2, compared with the index for the location path `/mondial/country`.

Figure 9 shows that access time slightly grows as the depth of PrefixSummary increases. It is obvious that increase in depth of the index enlarges the size of the index, as shown in Table 3, and thus access time of the stream also increases for all queries.

In the last experiment, we evaluate impact of the replication of PrefixSummary on the access time performance of the proposed scheme. For the given XML documents, we have generated broadcast streams that have different numbers of replicas of PrefixSummary within a broadcast cycle varying from 1 to 30. Figure 10 shows access time results of evaluating test queries on the SL method based stream. We can observe that as the number of index replicas grows in the stream, access time of the queries on the SIGMOD Record and Reed data tends to decrease until it reaches a minimum value and then increases very slightly. Particularly, a pulse curve occurs in the graph of Reed data for the following reason. If the tune-in point of the stream and a replica of PrefixSummary precede the target document, the access time can be reduced by the period of one broadcast cycle. However, if the replica of PrefixSummary nearest from the tune-in point appears after the first bucket of the target document, the access time is increased by the waiting time for the next broadcast cycle. This results in the pulse curve in Fig. 10. Note that the minimal access time of those
queries is almost the same as the access time on the stream without PrefixSummary, shown in the first experiment (See Fig. 7). This access time improvement is due to reduction in wait time for PrefixSummary from the tune-in point, as shown in Fig. 4. Meanwhile, queries M1 and M2 on the Mondial data have achieved no meaningful gain in access time from index replication. This is because Mondial data appears first in the stream and thus all replicas of PrefixSummary within a broadcast cycle have entries pointing to the data which will be delivered in the next broadcast cycle.

Note that index replication does not affect tuning time of the stream since probing of only one PrefixSummary is enough to evaluate the given query regardless of the number of its replicas.

6. Conclusions

In this paper we proposed a new index structure for the wireless stream of heterogeneous XML data to enable energy-efficient processing of path queries at the mobile client. We defined the PrefixSummary index which stores the address of a bucket in the stream containing the first XML node satisfying each location path in the XML data. The proposed index does not depend on the method to construct a stream of XML data, and thus can be used with any streaming method for XML. We presented algorithms to generate broadcast stream with PrefixSummary and to process a path query on the stream efficiently by exploiting the index information. We analyzed the latency and energy efficiency of the proposed approach in comparison with the conventional stream organization without any global index. We also conducted experiments using real XML data sets to evaluate the access time and tuning time performances of the proposed scheme. The experimental results show that our approach based on PrefixSummary can reduce tuning time for processing path queries significantly. This is mainly due to the immediate access to the first answer node of the given query in the stream of target documents by exploiting the pre-calculated address in the PrefixSummary index. Adding the proposed index may increase access time of the stream, but it can be also reduced by means of replicating the index over the broadcast stream.

Acknowledgments

This work was supported by the Dongduk Women’s University grant.

References

[1] S. Acharya, R. Alonso, M. Franklin, and S. Zdonik, “Broadcast disks: Data management for asymmetric communication environments,” Proc. ACM SIGMOD Conference, pp.199–210, 1995.
[2] A. Berglund, S. Boug, D. Chamberlin, M.F. Fernandez, M. Kay, J. Robie, and J. Simeon, XML Path Language (XPath) 2.0 Second ed., W3C Recommendation, 2010.
[3] S. Boug, et al., XQuery 1.0: An XML Query Language Second Edition, W3C Recommendation, 2010.
[4] T. Bray, J. Paoli, C.M. Sperberg-McQueen, E. Maler, and F. Yergeau, Extensible Markup Language (XML) 1.0 Fifth ed., W3C Recommendation, 2008.
[5] Y.D. Chung and M.H. Kim, “An index replication scheme for wireless data broadcasting,” J. Systems and Software, vol.51, no.3, pp.191–199, 2000.
[6] Y.D. Chung, S. Yoo, and M.H. Kim, “Energy- and latency-efficient processing of full-text searches on a wireless broadcast stream,” IEEE Trans. Knowl. Data Eng., vol.22, no.2, pp.207–218, 2010.
[7] ETSI EN 302 304 V1.1.1, “Digital Video Broadcasting (DVB); Transmission system for handheld terminals (DVB-H),” European Telecommunications Standards Institute, 2004.
[8] T. Imielinski, S. Viswanathan, and B. Badrinath, “Data on air: Organization and access,” IEEE Trans. Knowl. Data Eng., vol.9, no.3, pp.353–372, 1997.
[9] T. Imielinski, S. Viswanathan, and B. Badrinath, “Energy efficient indexing on air,” Proc. ACM SIGMOD Conference, pp.25–36, 1994.
[10] ISO/TS 24530-1:2006, “Traffic and Travel Information (TTI) via Transport Protocol Experts Group (TPEG) Extensible Markup Language (XML) - Part 1: Introduction, common data types and tpegML,” International Organization for Standardization, 2006.
[11] C.-S. Park, C.S. Kim, and Y.D. Chung, “Efficient stream organization for wireless broadcasting of XML data,” Proc. Asian Computing Science Conference, pp.223–235, 2005.
[12] J.P. Park, C.-S. Park, and Y.D. Chung, “Energy and latency efficient access of wireless XML stream,” J. Database Management, vol.21, no.1, pp.58–79, 2010.
[13] WiMAX Forum, http://www.wimaxforum.org
[14] XML Data Repository, http://www.cs.washington.edu/research/xml datasets.
[15] 3GPP Long Term Evolution, http://www.3gpp.org
Chang-Sup Park received his B.S., M.S., and Ph.D. degrees in Computer Science from KAIST, Daejeon, Korea, in 1995, 1997, and 2002, respectively. From 2002 to 2005, he worked for the R&D Center of KT Corporation in Seoul, Korea. In 2005, he joined the faculty of the University of Suwon, Korea. He joined the Department of Computer Science at Dongduk Women’s University in 2009, where currently he is an assistant professor. His research interests include wireless information systems, XML data management, semantic Web, and information retrieval. He is a member of the KIISE.

Jun Pyo Park received his B.S. degree in Computer Science from Dongguk University, Seoul, Korea, in 2005, and M.S. degree in Computer Science from Korea University, Seoul, Korea, in 2009. In 2009, he entered the Ph.D. course of Computer Science at Korea University, where currently he is a Ph.D. candidate. His research interests include database privacy, XML query processing, streamed data, wireless information retrieval, wireless broadcast systems, and mobile computing.

Yon Dohn Chung received his B.S. degrees in Computer Science from Korea University, Seoul, Korea, in 1994, and M.S. and Ph.D. degrees in Computer Science from KAIST, Daejeon, Korea, in 1996 and 2000, respectively. In 2006, he joined the faculty of the Department of Computer Science and Engineering, Korea University, Seoul, Korea. His research interests include broadcast databases, cloud computing, graph databases, XML databases, mobile computing and database systems. He is a member of the ACM, IEEE computer Society, and KIISE.