Efficient Cache Invalidation Schemes for XML Data Accesses in Mobile Environments

Po-Jen Chuang and Yu-Shian Chiu
Department of Electrical Engineering
Tamkang University
Tamsui, Taipei County
Taiwan 25137, R. O. C.
E-mail: pjchuang@ee.tku.edu.tw

Abstract- *Application of the eXtensible Markup Language (XML), a well-known standard serialized format on databases, in the World Wide Web (WWW) is growing swiftly in recent years. Previous researches on XML focus mainly on indexing techniques and broadcasting approaches, less on the updating, of the XML databases. In this paper, we apply ABI+HCQU and SWRCC+MUVI, two of our previously proposed cache invalidation schemes, to perform data updates in the XML databases. Our center design is to figure out the correlation between the XML databases and use it to assist data updates under the operation of the two cache invalidation schemes. Simulation results show that by making use of data correlation, our schemes attain higher cache hit ratios and lower query uplink ratios, which together yield much shorter average access time, than other related schemes.*

Keywords: *Mobile Environments, Cache Invalidation, eXtensible Markup Language (XML), Data Accesses, Data Updates, Data Correlation, Invalidation Reports, Experimental Performance Evaluation and Comparison.*

1. Introduction

A wireless network, such as a Notebook or PDA, usually consists of one or more servers and lots of clients. Databases are stored in the servers and the connection between the servers is wired. As a client needs to comply with the mobility and lightweight characteristics of a wireless network, its storage is limited and will store only a certain portion of the database. In such a wireless environment, if each client is installed with a cache and allowed to fetch the newly accessed hot data from the cache upon a query request, both the query uplink bandwidth and average data access time will be significantly reduced. In doing so, how to maintain data consistency between a client's cache and the server's database becomes an important and critical issue which has drawn considerable investigations from researchers, e.g., [1, 2, 3, 4].

In fact, the rapid advancement of mobile techniques has facilitated the application of mobile networks in quite a variety of daily uses, including weather prediction, traffic information, medical diagnosis and even military detection in recent years. Data gathered by mobile networks can be stored by XML [5] and accessed through XQuery [6]. The database can be managed using XQUF[7] and the Document Type Definition (DTD) for each XML document can be attained using [8]. Current researches on XML center either on the indexing techniques ([9, 10]) or on how to broadcast the XML data efficiently [11]. Updates on databases, by contrast, are less considered. Seeing this, [6] and [12] put their efforts to deal with the issue (i.e., updating the XML databases) to pursue more desirable function of XML.

General cache invalidation schemes for the wireless networks hardly consider the correlation between data. But in the XML databases, correlations among data do exist due to their hierarchical architectures, and if we ignore such data correlations and simply bring in general cache invalidation schemes to work on XML databases, performance will certainly degrade. In order to take the correlations among XML data into consideration and use, [13] broadcasts additional Structural Invalidation Reports (SIR) to update the architecture information in the cache. That is, such additional structure information must be broadcast if a general cache invalidation scheme is to be applied on the XML databases.

The main goal of this research is to define and use data correlations which exist in the XML databases to further investigate the update issue. To carry out our investigation, two of our previous cache invalidation schemes, ABI+HCQU and SWRCC+MUVI [14], are applied to work on the XML databases. In order to apply our schemes on the XML databases, we also broadcast the additional SIRs in [13] to manage the data correlations in the database. Simulation results show that by making use of data correlations (reserving the cache data with only architectural changes), our schemes attain higher cache hit ratios and lower query uplink ratios, and the two together

yield much shorter average access time, than the other cache invalidation schemes.

The paper is organized as follows. Section 2 gives background study on XML and cache invalidation. Section 3 describes the two cache invalidation schemes of ours and their applications on XML data. Experimental performance evaluation and comparison are provided in Section 4. Section 5 concludes the paper.

2. Background Study

2.1 The Extensible Markup Language (XML)

Elements are the basic components of the eXtensible Markup Language (XML). An element in XML may appear in any depth and contains any amount of sub-elements. Each element has two important tags, the start tag and end tag, to indicate its start and end. In this paper, we construct an XML data tree based on the One-Sibling-Address (OSA) [15, 10]. The following are its associated definitions.

Definition 1：Constructing the XML data with the tree architecture

1. **NAME**: Storing the node's name, it can be the Element plus Attribute or the Parsed Character DATA (PCDATA).
2. **DATA**: Storing the node's data, its data type is usually in strings or numbers.
3. **Child** (C): The child pointer of a node will store the address of its first child (its other children will be cascaded by this first child) or be set as null when the node has no children at all.
4. **Father** (F): The father pointer of a node will store the address of its father (or set as null if it has no father).
5. **Next** (N): The next pointer of a node stores the address of its next brother (or set as null when there is no next brother).
6. **Previous** (P): The previous pointer of a node stores the address of its previous brother (or set as null when there is no previous brother).

Definition 2：Constructing the Prefix Sequence Number (PSN) and the XML Path (XP)

1. **PSN**：The sequence number of each node obtained by tracing the XML data tree in the prefix order.
2. **XP**：All the visited nodes from the root to a node are included in the node's XP.

Each node in XML tree includes six information items which are defined by Definition 1. The path and sequence number of each node can be obtained by Definition 2, and the tree root is set as the first element/attribute in an XML file. For instance, for Mondial.xml [16, 17], which contains 34948 nodes, the sequence numbers are from 1~34948, while for XMark.xml [18, 19] (scaling factor = 0.1), which has 267438 nodes, the sequence numbers will be 1~267438.

2.2 The Update Operations

Our investigation in this paper sets up several update situations according to [12] and [13].

1. **Rename (ptr, name)**: Update only the element's NAME, not C, N, F, P and DATA. The cached XP will also be updated.
2. **Replace (ptr, content)**: Update only the node's DATA, not C, N, F, P, NAME or XP.
3. **Insert (ptr, newname, content)**: ptr is a pointer that points to a node in the XML tree. The Insert instruction will insert a new node (NAME="newname", DATA ="content") between a node that ptr points to and its Child.
4. **Delete (ptr)**: ptr is a pointer that points to a node (in the XML tree) whose descendants will be deleted (the Child pointer of the node will be set as null after deletion). If the Child pointer of the node that ptr points to is null before deletion, this Delete (ptr) becomes an invalid case.

Note that XP will change with Rename, Insert and Delete instructions but not with the Replace update, and we will update PSN and XP accordingly.

2.3 The TS (IR-Naïve) and XIR Schemes

2.3.1. The TS (IR-Naïve) Scheme [2]. The basic cache invalidation scheme is the TS scheme in [2], to be called as the IR-Naïve scheme in our later discussion because it broadcasts periodic invalidation reports (IR). At the end of each broadcast interval L, IR-Naïve will collect and put the IDs and timestamps of all data items that have been updated from T-w*L (w being the window size) to the current timestamp T on the IR. The server then downlinks the IR to all clients for them to invalidate their caches.

2.3.2. The XIR Scheme [13]. Two concepts, the influenced area (IA) and the node sequence information (NSI), are defined in [13] in order to implement cache invalidation schemes on the XML database.

XIR has two different channels, the broadcast channel (for the server to broadcast IRs and downlink the requested data) and the on-demand channel (for a client to uplink queries to the server). In a cache invalidation scheme, the server will periodically broadcast XIR which contains two IRs: the SIR and the data invalidation report (DIR). The SIR includes the updated NSI in XML and the corresponding DIR pointer. DIR includes IDs and timestamps which will update time from T_i-w*L to T_i. A client will update the XP of its cached data according to

SIR which has a pointer pointing to the corresponding DIR.

3. Our Proposed Schemes

To attain more desirable utilization of the wireless resources, we have in our previous researches divided data into different groups based on their frequencies of being updated or queried, and adjusted their broadcast intervals accordingly to fit the real demands.

If existing cache invalidation schemes are applied on the XML database without any modification, it is very likely that some still valid cached data items will be unnecessarily invalidated because the changed correlation between data will change the database architecture. To improve the situation, one can broadcast additional SIR. The implementation of IR and additional SIR has been illustrated in the XIR scheme in [13].

As mentioned before, the main purpose of this paper is to implement our ABI+HCQU and SWRCC+MUVI cache invalidation schemes [14] in the XML database and, by utilizing the data correlations in the XML database, to obtain efficient update of cached data. The detailed operation and implementation are given below.

● **The Adaptive Broadcasting Interval (ABI) Approach.**

To operate ABI, we first define a threshold H to decide if the information in an IR is valid. If the number of updated data exceeds H during 1 IR broadcasting interval L, the information in the IR is considered outdated. L needs to be adjusted: If the original L is $B1$, it can be changed to a shorter interval, say $B1/2$, to satisfy the frequent query requests of the clients. On the contrary, if the number of updated data falls below H during 1 L (i.e., the clients do not frequently file queries to the server), the length of L can be doubled ($B1\times2$) to save resources.

●**The Hot/Cold vs. Query/Update (HCQU) Approach.**

The HCQU approach divides data into 5 groups: **Hot Update (HU)**, **Hot Query (HQ)**, **Cold Update (CU)**, **Cold Query (CQ)** and the **ReMAinder (RMA)**, and assigns each group a proper broadcasting interval to suit the actaul demands. It operates as follows.

(1) The server and clients will predefine the five groups (to be redefined after a fixed time interval):

HU/HQ：The top 5% of data which are updated / queried most frequently .

CU/CQ：The bottom 5% of data which are updated / queried least frequently.

RMA：The remainder of data, belonging to none of HU, HQ, CU or CQ.

(2) The server will broadcast 5 different IRs, HU-IR, HQ-IR, CU-IR, CQ-IR and RMA-IR, in their own broadcasting intervals.

(3) Repeat steps 1 and 2.

Algorithm 1: The Algorithm at the Server

(A) At the end of each corresponding broadcast intervals: *HU-L / HQ-L / CU-L / CQ-L / RMA-L*,
  Construct the corresponding IR_i: HU-IR_i / HQ-IR_i / CU-IR_i / CQ-IR_i / RMA-IR_i;
  Broadcast the corresponding IR_i and requested data: L_{bcast};
  Requested data =φ;

(B) Receive a request L_{data} from client C_j :
  $L_{bcast} = L_{data} \cup L_{bcast}$;

Algorithm 2: The Algorithm at the Clients

(A) When a client C_j receives the corresponding IR_i and L_{bcast}:
  if (Tl<T_i –w* corresponding-L),
    Drop the corresponding group (HU/HQ/CU/CQ/RMA) data entry stored in the cache;
  else {
  for each data item in the cache do
    if (the structural information is updated)
      Invalidate the structural information;
      if (the data information is updated)
        Invalidate the cached XML data;
  for each L_{bcast} do
    if (L_{data} is the requested data)
      then download;
      Use the data to answer the previous request;
  }
  Save the last received corresponding IR_i timestamp as the corresponding timestamp
  if (a request is issued)
    query(Q)

(B) Procedure query(Q)
  When a client issues a query Q;
  if (the structural information of the requested data is invalidated)
    Check the related data information;
    if (the related data information is valid)
      $L_{data} = L_{data} \cup$ the structural information
    else
      $L_{data} = L_{data} \cup$ the entire requested XML data;
  else use the cached XML data
  for each XML data $\in$ L_{data} send request (L_{data})

●**ABI+HCQU.**

After classifying the database into the five groups, we use the ABI to assign each data group a suitable broadcast interval to obtain efficient broadcasting. The broadcast algorithm for the server is given in Algorithm 1, and the cache invalidation algorithm at the clients is given in

Algorithm 2. The five broadcast intervals in ABI+HCQU, i.e., *HU-L, HQ-L, CU-L, CQ-L* and *RMA-L*, are called the corresponding broadcast intervals, and the five IRs, i.e., HU-IR, HQ-IR, CU-IR, CQ-IR and RMA-IR, are the corresponding IRs. Each corresponding IR has its own corresponding timestamp and includes both the DIR and SIR, as defined in Section 2. Note that the SIR is the additional broadcast information which is absent in our original ABI+HCQU cache invalidation scheme but added here to suit the XML database.

● **The Sleep/Wakeup/Recovery/Check /Confirm (SWRCC) Approach.**

As a mobile client may get disconnected from the server actively (to conserve energy) or passively (due to moving or accidental factors), we thus need different cache invalidation approaches to verify the validity of the client's cached data after reconnection. In our SWRCC approach, a client will send a ***Sleep*** message to inform the server of its intended disconnection. When the client reconnects to the server and receives the first query, it will send a ***Wakeup*** message (including also the information of the received first query) to the server. Receiving such a Wakeup message, the server in turn sends the client a ***Recovery*** message which encloses all data items that have been updated during the client's disconnection and also the valid information of the first query. The client then invalidates its cache by the received Recovery message.

When a client gets reconnected to the server and receives a query request after passive disconnection, it will send a ***Check*** message to the neighboring clients (via low-power broadcasting) at every Check Period time interval (to be set as 1/4 or 1/5 of the IR broadcasting interval). The Check message includes the id set of queries received during the time interval. Upon receiving such a Check message, a neighboring client first checks its own cache for the requested data item. If the item is in its cache and is still valid, the neighboring client then sends a ***Confirm*** message to the reconnected client (who will not answer the query until receiving the next IR from the server) and also the other neighboring clients.

● **Modified/Uncertain/Valid/Invalid (MUVI): The Validity States.**

To indicate the validity of cached items, we use two additional bits to denote the state of each cached item as **Modified**, **Uncertain**, **Valid** or **Invalid**. When a client reconnects to the server and receives a query, it will send a Check message to the neighboring clients and receive possibly valid data from their Confirm messages. At this point, the state of these possibly valid cached data will be changed from Uncertain to **Modified (M)**. When a client is initially reconnected to the server, the validity of its cached data is uncertain and is therefore marked as **Uncertain (U)**. After reconnection and receiving the first IR, the client moves on to check the validity of its own cached data. The state of data items which are still valid is marked as **Valid (V)**, while the state of data items which become invalid is changed into **Invalid (I)**. The detail concept of MUVI is in [14].

Algorithm 3: The Algorithm at the Server

(A) At the end of each broadcast interval L:
  Construct IR_i;
  Broadcast IR_i and the requested data: L_{bcast};
  Requested data = φ;
(B) Receive a request L_{data} from client C_j:
  $L_{bcast} = L_{data} \cup L_{bcast}$;
(C) Receive a **Sleep** message from the client
  Store the disconnection timestamp of the client.
(D) Receive a **Wakeup** message from the client
  Broadcast the **Recovery** message that includes the data updated between **Sleep** and **Wakeup**

Algorithm 4: The Algorithm at the Clients

(A) Before going into active disconnection,
  The client sends a **Sleep** message to the server.
(B) When coming back from active disconnection,
  The client sends a **Wakeup** message to the server;
  Invalidates its cache by the **Recovery** message received from the server.
(C) **if** (cache misses or cache hits but Uncertain in the client's cache)
  Broadcasts the **Check** message to the neighbors:
  if (receives the **Confirm** message during the Check Period Time Interval)
  then invalidates its own cache and waits for the next IR
  else query (Q)
(D) When a neighbor client receives the **Check** message
  if (cache hits)
  then the neighbor client sends a **Confirm** message to the **Check** message sender.
(E) When a client C_j receives IR_i and L_{bcast}:
(F) Procedure query (Q):
  (Steps E and F are the same as in algorithm 2.)

●**SWRCC+MUVI.**

We believe SWRCC+MUVI can be implemented into the XIR scheme [13] to enhance its ability to handle the disconnection consequences. Algorithms 3 and 4 demonstrate respectively from the server's and client's sides how to effectuate SWRCC+MUVI in the XML database. Based on the algorithms, the server will construct an IR and broadcast it at the end of each broadcast interval. The broadcast IR (i.e., XIR) will include DIR and SIR as defined in Section 2. Upon receiving the XIR, a client will invalidate its cache according to the structural and data information in it. (Note that SIR is the additional broadcast information

which is absent in our original SWRCC+MUVI cache invalidation scheme and is added here to comply with the distinct features of the XML database.)

4. Simulation Results

4.1 Simulation Environment

Our simulation environment is similar to that in [13, 2, 3], with some necessary changes to comply with the XML database. The adopted simulation parameters are as follows. The database size = the number of tree nodes (DB = 34948 in Mondial.xml and 267438 in XMark.xml); the cache size = 1/100 ~ 1/10 of the database, with the default value = 1000. The mean inter-update time is 100 sec and the mean inter-query time is 30 sec -- both are of exponential distribution. The update or query probability is of Zipf distribution: The i_{th} data has a probability of$1/ i^z$ to update or query, z = 0.95 [20]. We meanwhile set the offset range [21] under the Zipf distribution from 0 to 80 -- a bigger offset indicates a less frequently updated queried item. There are four types of updates: Replace, Rename, Insert and Delete, whose probabilities are 0.4, 0.4, 0.1 and 0.1 respectively.

The mean inter-disconnection time is 100 sec; each disconnect period is 50~5000sec. The broadcast interval L = 20 sec and the broadcast window size w = 10. Either the uplink bandwidth, downlink bandwidth or bandwidth between clients is set as 10 Kbps. We set the first 5% of the database as the HOT data while the rest as the COLD and RMA data. There are one server and 10 clients in the model. The timestamp size = 32 bits, the data ID size = 332 bytes (because the maximum path length in Mondial.xml and XMark.xml is 332 bytes) and the data item size = 67 bytes (the average data size is 34 bytes in Mondial.xml and 67 bytes in XMark.xml).

4.2 Simulation Results

A number of performance parameters, such as the cache hit ratio, the query uplink ratio and the average access time, are simulated with different values of Offset or various disconnection probabilities under benchmarks Mondial.xml and XMark.xml, and the results are collected to evaluate and compare the performance of our cache invalidation schemes (ABI+HCQU and SWRCC+MUVI) and other schemes (XIR and IR-Naïve) in updating the XML databases.

The following are our definitions for the cache hit ratio, the query uplink ratio and the average access time.

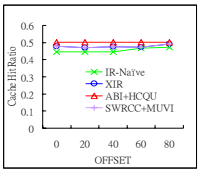
The cache hit ratio = (# of queried data which are "in cache & valid")/(# of queries)

The query uplink ratio = (# of queried data which are "in cache but invalid" or "not in cache")/(# of queries)

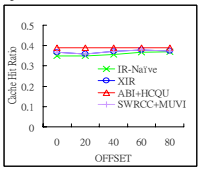
The average access time = (the waiting time + data transmission time)/(# of queries)

4.2.1. The Cache Hit Ratio. Figure 1 shows that IR-Naïve, which fails to consider data correlations, generates lower cache hit ratios than XIR, SWRCC+MUVI and ABI+HCQU which all yield better cache hit ratios due to the broadcasting of additional architecture information. The bigger OFFSET, the more rare update data will be queried frequently. The smaller OFFSET, the popular update data will be queried frequently. When the cache hit ratio grows, both the query uplink ratio and average access time get reduced.

Cache hit ratios with various disconnection probabilities are plotted in Figure 2. As SWRCC+MUVI allows clients to request data (when possible) from the neighboring nodes after reconnection, it is least influenced by the effect of disconnection.

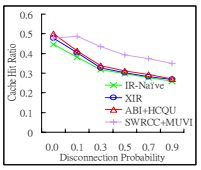


(a) Mondial.xml

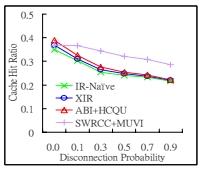


(b) XMark.xml

Figure 1. The cache hit ratios with different values of Offset.

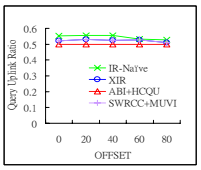


(a) Mondial.xml

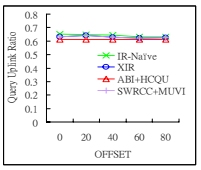


(b) XMark.xml

Figure 2. The cache hit ratios with various disconnection probabilities.

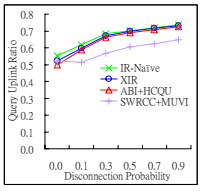


(a) Mondial.xml

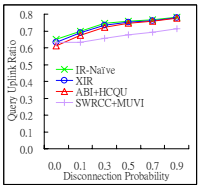


(b) XMark.xml

Figure 3. The query uplink ratios with different values of Offset.

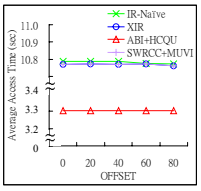


(a) Mondial.xml

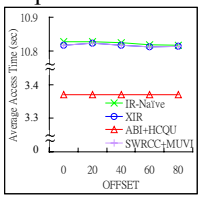


(b) XMark.xml

Figure 4. The query uplink ratios with various disconnection probabilities.

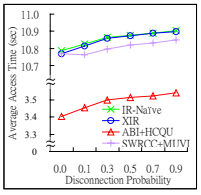


(a) Mondial.xml

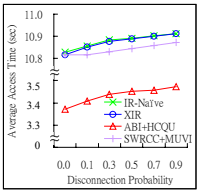


(b) XMark.xml

Figure 5. The average access time with different Offset values.



(a) Mondial.xml



(b) XMark.xml

Figure 6. The average access time with various disconnection probabilities.

4.2.2. The Query Uplink Ratio. In Figure 3, among the four schemes, our ABI+HCQU yields the lowest query uplink ratios thanks to its data classification and adaptive broadcast intervals which effectively help a client reduce its necessity to uplink requests to the server (and as a result takes the least time to get the requested data). The additional broadcast architecture information of XIR and SWRCC+MUVI also helps the two schemes attain lower query uplink ratios than IR-Naïve.

Figure 4 shows that SWRCC+MUVI stands out as the one scheme least affected by the disconnection consequences. This is because after reconnection it is able to request data directly from neighbors, when available.

4.2.3. The Average Access Time. Without disconnection, as Figure 5 displays, the average access time for SWRCC+MUVI and XIR appear similar. For IR-Naive, which has the lowest cache hit ratio (due to lack of architecture information) and therefore higher query uplink ratios, the average access time gets increased too.

Our SWRCC+MUVI scheme, which is specifically designed to reduce the disconnection consequences, reassures its advantage again in Figure 6: It yields an overall average access time that is least affected by different disconnection probabilities.

The two figures both clearly demonstrate a unique result: Due to its data classification and adaptive broadcast intervals, our ABI+HCQU approach generates distinctively lower average access time than the other schemes in all situations.

5. Conclusion

Traditional cache invalidation schemes for the wireless networks hardly deal with the correlations between data. In the XML database architecture, correlations nevertheless exist among the entire database because the data items are cascaded. To attain more desirable data updates using the data correlations in the XML databases, we apply two of our previous cache invalidation schemes to work in the XML data structure with certain modifications. The major modification is to broadcast additional SIRs (which are absent in our original schemes) to manage the data correlations in the XML database, and it requires only some architectural change to do so. Simulation results show that our ABI+HCQU scheme can produce much shorter average data access time than other related schemes. The results also show that when either active or passive disconnection happens, our SWRCC+MUVI scheme outperforms other schemes in countering the disconnection consequences. Both schemes show that they are able to attain favorable data updates for the XML database and meanwhile conserve the restricted energy and bandwidth resources.

References

[1] J.H. Choi and S.K. Lee, "An Efficient Cache Access Protocol in a Mobile Computing Environment," *Proc. 3rd Int'l Conf. on ISPA*, Nov. 2005, pp.1123-1134.

[2] D. Barbara and T. Imielinski, "Sleepers and workaholics: caching strategies in mobile environments," *Proc. 1994 ACM SIGMOD Conf. on Management of Data*, May. 1994, vol. 23, no. 2, pp.1-12.

[3] G. Cao, "A scalable low-latency cache invalidation strategy for mobile environments," *IEEE Transaction on Knowledge and Data Engineering*, vol. 15, no.5, pp.1251-1265, 2003.

[4] A. Madhukar and R. Alhajj, "An adaptive energy efficient cache invalidation scheme for mobile databases," *Proc. 21th ACM SAC*, Apr. 2006, pp.1122-1126.

[5] http://www.w3.org/XML/

[6] http://www.w3.org/TR/xquery/

[7] http://www.w3.org/TR/xquery-update-10/

[8] http://saxon.sourceforge.net/dtdgen.html

[9] J. Lu, T. W. Ling, C.-Y. Chan, and T. Chen, "From region encoding to extended dewey: On efficient processing of xml twig pattern matching," *Proc. 31st Int'l Conf. on VLDB*, Aug. 2005, pp.193-204.

[10] Y.D. Chung, and J.Y. Lee, "An indexing method for wireless broadcast XML data," *Information Sciences: an International Journal*, vol. 177 no. 9, pp.1931-1953, May. 2007.

[11] S.H. Park, J.H. Choi, and S. Lee, "An effective, efficient xml data broadcasting method in a mobile wireless network," *Proc. 17th Int'l Conf. on DEXA'06*, Sep. 2006, pp.358-367.

[12] I. Tatarinov, Z. G. Ives, A.Y. Halevy, and D. S. Weld, "Updating XML," *Proc. 2001 ACM SIGMOD*, June 2001, pp.413-424.

[13] J. H. Choi, S. H. Park, M. S. Lee, Y. D. Chung, and S. K. Lee, "XIR: cache invalidation strategy for xml data in mobile environments," *Proc. 2007 ACM MobiDE*, June 2007, pp.79-82.

[14] P.-J. Chuang and Y.-S. Chiu, "Constructing efficient cache invalidation schemes in mobile environments," *Proc. 3rd Int'l Conf. on Signal Image Technology & Internet Based Systems*, Dec. 2007, pp.260-267.

[15] C.S. Park, C.S. Kim and Y.D. Chung, "Efficient stream organization for wireless broadcasting of xml data," *Proc. 2005 ASIAN*, Dec. 2005, pp.223-235.

[16] http://www.informatik.uni-freiburg.de/~may/mondial/florid-mondial.html

[17] http://www.cs.washington.edu/research/xmldatasets/www/repository.html

[18] A. Schmidt, F. Waas, M.L. Kersten, M.J. Carey, I. Manolescu, and R. Busse, "Xmark: A benchmark for xml data management," *Proc. 28th Int'l Conf. on VLDB*, Aug. 2002, pp.974-985.

[19] http://monetdb.cwi.nl/xml/generator.html

[20] B. Mandhani and D. Suciu, "Query caching and view selection for XML databases," *Proc. 31st Int'l Conf. on VLDB*, Aug. 2002, pp.469-480.

[21] S. Acharya, M.J. Franklin, and S.B. Zdonik. "Disseminating updates on broadcast disks," *Proc. 22nd Int'l Conf. on VLDB*, Sep. 1996, pp.354-365.