Efficient Multichannel in XML Wireless Broadcast Stream

Arezoo Khatibi*1 and Omid Khatibi2

1Faculty of Computer Science, University of Kashan, Kashan, Iran
2Faculty of Mathematics, University of Vienna, Vienna, Austria

Abstract—In this paper we recommend the use of multichannel for XML data in wireless broadcasting. First we divide XML data into information units as bucket, then extract path information (XPath) for any unit and build an index tree from the data path. Finally, make wireless data stream with merging parts of index tree and parts of XML data in multichannel XML. Then, create a protocol that allows mobile users access to the wireless XML stream generated with our method. We study 11 channels in server side and 3 orthogonal channels in client side.

Index Terms—Keywords: XML indexing; Multichannel; Wireless broadcasting; Mobile databases; Wireless information systems.

I. INTRODUCTION

A Sample of an XML stream is:

```xml
<Root><a1><b1><c1/></b1><c2/><c3/></a1><b2><c4><c5><c6/></c5><c7/></c6><b3><c8><c9/></c8><a1/></b3></b2>
<b3><c10><c11/></c10><c12/></c12><a2><a3></a2><a3></a3></a2><b4><c14><c15></c15><b5><c16><c17><c18/></c18><b6><a2><a3></a2><a3></a3></a2><b7><c19><c20/></c20><b7><b8><c21/></b7><b8><b8><c22><c23><c24/></c24><b8><b9><c25/></b9><c25/></b9><b9><b9><c26><c27/></c27><b9><a3><a3></a3></Root></c26></b9></a3></c26></b9></b9></c22></c23></c24></b8></b8></b7></b6><b16><b17><b18></b18><b6><a2><a3></a2><a3></a3></a2><b7><c19><c20/></c20><b7><b8><c21/></b7><b8><b8><c22><c23><c24/></c24><b8><b9><c25/></b9><c25/></b9><b9><b9><c26><c27/></c27><b9><a3><a3></a3></Root></c26></b9></a3></c26></b9></b9></c22></c23></c24></b8></b8></b7></b6><b16><b17><b18></b18><b6><a2><a3></a2><a3></a3></a2><b7><c19><c20/></c20><b7><b8><c21/></b7><b8><b8><c22><c23><c24/></c24><b8><b9><c25/></b9><c25/></b9><b9><b9><c26><c27/></c27><b9><a3><a3></a3></Root></c26></b9></a3></c26></b9></b9></c22></c23></c24></b8></b8></b7></b6><b16><b17><b18></b18><b6><a2><a3></a2><a3></a3></a2><b7><c19><c20/></c20><b7><b8><c21/></b7><b8><b8><c22><c23><c24/></c24><b8><b9><c25/></b9><c25/></b9><b9><b9><c26><c27/></c27><b9><a3><a3></a3></Root></c26></b9></a3></c26></b9></b9></c22></c23></c24></b8></b8></b7></b6></Root>
```

And tree presentation from this XML stream is in the figure 1. Yon Dohn Chung and Ji Leon Lee make a full study of an indexing for wireless broadcast XML data[1]. We develop it by using multichannel and multiradio. Data broadcast is widely used because of bandwidth limitations of wireless systems, where the server broadcasts data through a distribution channel and then the customers listen to the channel [3]. Customer can retrieve relevant information, furthermore, the client does not need to send a request to the server [4]. In order to retrieve information, we have an index broadcasting direct information of Published data items in server [4]. In order to retrieve information, we have an index broadcasting direct information of Published data items in server side [4]. In order to retrieve information, we have an index broadcasting direct information of Published data items in server side [4]. In order to retrieve information, we have an index broadcasting direct information of Published data items in server side [4].

B. define tp strategy

Tp stands for the tree of index and path of data. This building strategy of the replication stream, replicates high level data path and high level index tree and put them before low level data path and low level index tree on the broadcast stream. It is suitable because of access time and tuning time for unique channel in compare of others method as pp, tt, (1, x).

III. OUR PROPOSED METHOD

We first implement, the XML tree traversal that we have noted in the introduction to the Java language along with the output of it.

```java
1. import java.io.File;
2. import java.util.LinkedList;
3. import javax.xml.parsers.DocumentBuilder;
4. import javax.xml.parsers.DocumentBuilderFactory;
5. import org.w3c.dom.Document;
6. import org.w3c.dom.Element;
7. import org.w3c.dom.Node;
8. import org.w3c.dom.NodeList;
9. /**
10. */
11. public class Main {
12. public static void main(String[] args) {
13. try {
14. DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
15. DocumentBuilder db = dbf.newDocumentBuilder();
```
16. File file = new File("C:\New\folder\c.xml");
17. if (file.exists()) {
    Document doc = db.parse(file);
    Node rootNode = doc.getFirstChild();
18. // System.out.println(rootNode.getNodeName());
19. // System.out.println(rootNode.
20. // getFirstChild().getNodeName());
21. // System.out.println(rootNode.
22. // getFirstChild().getNextSibling().getNodeName());
23. System.out.println(rootNode.
24. // node.getNodeName());
25. // node.getFirstChild();
26. // node.getNextSibling();
27. System.out.println(getXML(rootNode));
28. for (int i = 0; i < rootNode.getChildNodes().getLength(); i++) {
        a += "<" + rootNode.getNodeName() + "">
29. System.out.println(myNode.getNodeName() + " index: " + ll.indexOf(myNode));
30. System.out.println();
31. if (myNode != root)
32.     System.out.println(myNode.getNodeName());
33. ll.add(myNode);
34. } else if (thisNode.getPreviousSibling() != null)
35. thisNode = thisNode.getPreviousSibling();
36. } else if (thisNode.getSibling() != null)
37. thisNode = thisNode.getSibling();
38. ll.add(thisNode);  
39. return ll;
40. }
41. Node thisNode = root.getFirstChild();
42. while (!thisNode.equals(root)) {
43. ll.add(thisNode);
44. if (thisNode.hasChildNodes()) {
45. thisNode = thisNode.getFirstChild();
46. } else if (thisNode.getNextSibling() != null)
47. thisNode = thisNode.getNextSibling();
48. } else {
49. return ll;
50. }
51. 
Fig. 1: tree presentation from first sample of XML stream."
Fig. 2: Sequence d-link, c-link, s-link, h-link are data node address, the most left child node, the next sibling node and the relevant homolog node.

| TABLE I: Show the segments in each channel of multichannel. |
|-------------------------------------------------------------|
| Ch1 | 1 | 4 | 7 | 1 | 4 |
| Ch2 | 2 | 5 | 8 | 2 | 5 |
| Ch3 | 3 | 6 | 9 | 3 | 6 |

segments 3, 6, 9 on channel 3 and as we know each segment consisted (HI+LI+HD+LD); then, in General, we can apply at most 3 orthogonal channels in client side and at most 11 channels in server side.

IV. PROTOCOL

We use protocol 802.11 in client for receiving information. Each mobile customer can listen to only three channels at a moment.

V. ANALYSIS

At first we want to utilize access time with our proposed method in TP strategy. In TP strategy HI replicate in form sub tree; therefore, average access time considered as follows: Where n equals to the length of a high level in an XML tree in one channel. We consider a full tree with H height and h referred as replication level and each node is in one bucket and do not consider download time after finding data. We assume value of index node in i level as linear function of f from i and k is a positive number, in our proposed method f(i)=k, we assume data node value in i level as g(i).

A. For the average access time calculating by using TP strategy in multichannel case

\[
\text{Avg}_{ATP} = \frac{1}{2} \left( \frac{n}{3} \right)^{h-1} + 1)k \left[ \left( \frac{n}{3} \right)^h - 1 \right] + \left( \frac{n}{3} \right)^{h-H} - 1 \right] + \frac{1}{2} \left( \frac{n}{3} \right)^h + 1 \right] w \left[ h + \left( \frac{n}{3} \right)^{h-H} - 1 \right].
\]

(V.1)

Fig. 3: In Shows access time of tp strategy in single channel.

TABLE II: Show the symbols guide using in formula.

| Mark | Guide |
|------|-------|
| \(\Delta LD\) | The total size of an LD |
| \(\Delta HD\) | The total size of an HD |
| \(\Delta LI\) | The total size of an LI |
| \(\Delta HI\) | The total size of an HI |
| \(\Delta INDEX\) | The size of total index in a broadcast stream |
| \(\Delta DATA\) | The amount of data in a broadcast stream |
| f(i) | The size of an index node in a level i |
| g(i) | The size of the data node in a level i |
| X | The number of index replication in a (1,X) method |
| n | The fan out of a XML data and index trees |
| H | The height of a XML data and index trees |
| h | The replication level |
| w | Positive constant |
| k | Positive constant |
TABLE III: Get value of the access time used tp strategy with specified values in multichannel.

| n  | h  | H | K | w | AvgAT_{TP}       |
|----|----|---|---|---|------------------|
| 6  | 4  | 7 | 3 | 30| Single channel= 107823 × 10^9 |
| 6  | 4  | 7 | 3 | 30| Multichannel=80784 |

Fig. 4: Shows access time of tp strategy in multichannel.

**B. For the index size calculating by using TP strategy in multichannel case**

\[
\Delta INDEX_{TP} = \left( \frac{n}{3} \right)^h (\Delta H_{TP} + \Delta L_{TP}) \\
= k \left( \frac{n}{3} \right)^h \left[ \left( \frac{2}{3} \right)^h - 1 \right] + \left( \frac{n}{3} \right)^h \left[ \left( \frac{2}{3} \right)^h \left( \frac{n}{3} \right)^{H-h} - 1 \right]
\] (V.2)

TABLE IV: Get value of the index used tp strategy with specified values in single channel and multichannel.

| n  | h  | H | K | \Delta INDEX_{TP} |
|----|----|---|---|------------------|
| 6  | 4  | 7 | 3 | Single channel=1174176 |
| 6  | 4  | 7 | 3 | Multichannel=1056 |

Fig. 6: Shows index size with tp strategy in single channel.

Fig. 7: Shows index size with tp strategy in multichannel.

Fig. 5: Comparison access time in single channel and multichannel.

Fig. 8: Comparison index size of tp strategy between single channel and multichannel.
C. For the average tuning time calculating by using TP strategy in multichannel case

\[
AvgTT_{TP} = 1 + \Delta HI_{TP} + \Delta LI_{TP}
= 1 + k\left[\left(\frac{h}{3}\right)^h - 1 + \left(\frac{h}{3}\right)^{h-H} - 1\right]
\]  
\text{(V.3)}

\[
\begin{array}{cccccc}
 n & h & H & K & AvgTT_{TP} \\
 6 & 4 & 7 & 3 & Single channel= 907 \\
 6 & 4 & 7 & 3 & Multichannel=67 \\
\end{array}
\]

\text{TABLE V: get value of the tuning time used tp strategy with specified values in multichannel.}

Fig. 9: tuning time in multichannel case.

Fig. 10: tuning time in single channel case.

Fig. 11: comparison tuning time between multichannel and single channel.

D. For the data size calculating by using TP strategy in multichannel case

\[
\Delta DATA_{TP} = \left(\frac{n}{3}\right)h(\Delta HD_{TP} + \Delta LD_{TP})
= \left(\frac{n}{3}\right)\sum_{i=1}^{h} g(i) + \sum_{i=1}^{h-H} g(i) \times \left(\frac{n}{3}\right)^{i-1}
\]
\[
= w\left(\frac{n}{3}\right)^k[H + \left(\frac{2}{3}\right)^{h-H} - 1]
\]  
\text{(V.4)}

\[
\begin{array}{cccccc}
 n & h & H & w & \Delta DATA_{TP} \\
 6 & 4 & 7 & 30 & Single channel= 1827360 \\
 6 & 4 & 7 & 30 & Multichannel=5280 \\
\end{array}
\]

\text{TABLE VI: get value of the data size used tp strategy with specified values in multichannel.}

Fig. 12: data size of tp strategy in single channel.
VI. CONCLUSION

With increasing number of channels, data is broadcast faster in air. In this model, we must have a multiradio or have single device with several radio. We observed that tuning time reduced with increasing number of channel or reducing number of nodes. In index size with increasing channel or decreasing number of nodes, index size increase faster and displayed more mutant. While accessing time in multichannel case with increasing replication level we see access time decreases faster. In future work we recommend a method to get rid of traverse tree structure in multichannel xml wireless broadcast. Nonfinancial and no fund.

REFERENCES

[1] Yon Dohn Chung, Ji Yeon lee, An indexing Method for Wireless broadcast xml data, Information Sciences, Vol. 177, (2007), 1931-1953.
[2] Wai Gen Yee, Student Member, Efficient Data Allocation Over Multiple Channels at Broadcast Servers, IEEE Transactions on Computers, Vol. 51, no 10, October (2002), 1231-1236.
[3] T. Imielinski, S. Viswanathan, and B.R. Badrinath, Data on Air: Organization and Access, IEEE Transactions on Knowledge and Data Engineering, Vol. 9, no 3, May/June (1997), 353-371.
[4] C C. Lee, Y. Leu, Data on Air: Organization and Access, Information Sciences, Vol. 171, (2005), 445-458.