21-11-2005

Collaborative editing using an XML protocol

S. J. Davis
University of Wollongong, stdavis@uow.edu.au

I. S. Burnett
University of Wollongong, ianb@uow.edu.au

Collaborative editing using an XML protocol

Abstract

XML is a popular approach to interoperable exchange of data between a wide range of devices. This paper explores the use of the Remote XML Exchange Protocol as a mechanism to provide efficient interaction with complex XML documents to users with limited complexity devices and/or limited bandwidth connections. The interactive mechanisms provided by the protocol allow users to navigate, edit and download XML even when delivery of the full XML document is impossible. The paper examines the use of the protocol to enable multiple users to collaboratively edit remote XML documents. Further, the paper explores the combination of the protocol, collaborative editing and recently released Word processor/Office suite XML schema formats.

Disciplines

Physical Sciences and Mathematics

Publication Details

This article was originally published as: Davis, S. J. & Burnett, I. S., Collaborative editing using an XML protocol, IEEE International Region 10 Conference (TENCON 2005), Melbourne, November 21-24 2005, 1-5.

Collaborative Editing using an XML Protocol

Stephen J Davis[1] and Ian S Burnett

School of Electrical, Computer and Telecommunications Engineering
University of Wollongong, Australia

Abstract— **XML is a popular approach to interoperable exchange of data between a wide range of devices. This paper explores the use of the Remote XML Exchange Protocol as a mechanism to provide efficient interaction with complex XML documents to users with limited complexity devices and/or limited bandwidth connections. The interactive mechanisms provided by the protocol allow users to navigate, edit and download XML even when delivery of the full XML document is impossible. The paper examines the use of the protocol to enable multiple users to collaboratively edit remote XML documents. Further, the paper explores the combination of the protocol, collaborative editing and recently released Word processor/Office suite XML schema formats.**

Index Terms—**XML, Protocol, Collaborative Editing**

I. Introduction

XML [1] has become increasingly popular for representing and exchanging data. However, whilst XML provides interoperability amongst devices, this comes at a cost of increasing the raw data transmitted; this is primarily caused by the process of surrounding data with tags. XML and XML Schema [2] are also now being used in office suites such as OpenOffice (which uses the Open Document Format for Office Applications (OpenDocument) standardized by OASIS [3]) and Microsoft Office [4]. Thus, there is increasing interest in mechanisms which allow all users to interact with XML documents.

Often, many authors contribute to the creation and editing of a document. One approach is to edit a file and email it to the next author(s) but there is always the danger of different versions of a document being edited at the same time. For users with limited bandwidth transmitting many revisions of very large documents is not even feasible and there are thus many reasons to consider collaborative editing solutions.

Collaborative editing solutions usually retain a central copy of a document such that authors can "update" their version before editing. Further time and bandwidth savings can be achieved, if only altered sections of documents are uploaded and users can download only the document sections of interest.

[1] Partially funded by the Smart Internet Technology CRC

```
<Media xmlns="mediaNS:2004">
  <Music>
    <Song id="0071">
      <Title>Hit.1</Title>
      <Description>Theme song</Description>
      <Artist>A. Artist</Artist>
      <Format>MP3</Format>
      <Length>02:23</Length>
    </Song>
    <Song id="0328">
      <Title>Hit.2</Title>
      <Description>Song 2</Description>
      <Artist>B. Artist</Artist>
      <Format>OGG</Format>
      <Length>03:46</Length>
    </Song>
  </Music>
</Media>
```

Fig. 1. Example XML

Techniques such as those described in [5] and MPEG-B [6] provide partial solutions to efficient collaborative editing of XML documents, but do not provide a detailed solution. Thus, in this paper, we demonstrate the use of a complete two way protocol known as Remote XML Exchange Protocol to allow devices to download and edit remote fragments of XML documents efficiently in both compressed and non-compressed form.

II. Remote XML Exchange Protocol

The Remote XML Exchange Protocol (RXEP) is designed to handle the underlying delivery of XML Fragments. MPEG-B provides a solution for delivery of XML fragments as part of its set of standard Binary tools for XML, however, the fragment size and delivery are decided purely by the sending device, with the client-side user having no control over transmissions. In contrast, RXEP extends the MPEG-B solution by allowing users to control which fragments are to be delivered, the fragment size, and the timing of fragment delivery. In general, this technique requires the server to "listen" to requests and dynamically create fragments tailored to each peer. However, caching of requests is a possible option.

Clients which implement RXEP requests are able to

```
<FRU>
   <XPath location="/Media/Music/Song[1]">
</FRU>
```

Fig. 2. Example RXEP FRU

```
<RXEP>
   <ADD location="/Media/Music /Song[1]">
         <Title>Hit.1</Title>
        <Description>Theme song</Description>
        <Artist>A. Artist</Artist>
        <Format>MP3</Format>
        <Length>02:23</Length>
   </ADD>
<RXEP>
```

Fig. 3. Example RXEP FUU result from the FRU as in Fig. 2

query and browse (navigate) through remote XML documents, retrieving only relevant document fragments. This introduces significant savings as it avoids the user retrieving the entire XML document if only a small section of that document was desired. RXEP commands are defined and implemented using XML Schema in two parts: upstream commands (RXEP Fragment Request Units) and downstream commands (RXEP Fragment Update Units).

A. *Fragment Request Units*

Fragment Request Units (FRUs) are created by the users to request fragments of XML from a remote XML document. The FRUs are created in XML (valid to the FRU Schema) from a selection of RXEP commands. Briefly, basic FRU commands are as follows:

- Get – this requests a remote XML Document to begin browsing/navigating (similar to a HTTP get request);
- XPath – Delivers an XPath expression to the remote server to evaluate on the XML document, and deliver back the results;
- XMLPull – Provides a node-by-node navigation interface to the client allowing navigation commands such as: Next, Up, Back and Expand; and
- Stream – Allows fragments (determined by the server) to be streamed to the client.

FRUs are capable of requesting any fragment (based on fragment size and location), thus providing clients with random access into the XML. This allows a client to jump into any node in an XML document, or to simply, "navigate backwards" if previous XML fragments have not been cached.. A sample RXEP FRU requesting the child nodes of the */Media/Music/Song[1]* node (see Fig. 1) is illustrated in Fig. 2.

B. *Fragment Update Units*

Fragment Update Units (FUUs) are commands in XML which instructs the client to update parts of an XML Document. FUUs define commands such as:

- Add – add the fragment to the specified location defined by an XPath locator;
- Delete – deleted the branch specified by the XPath locator;

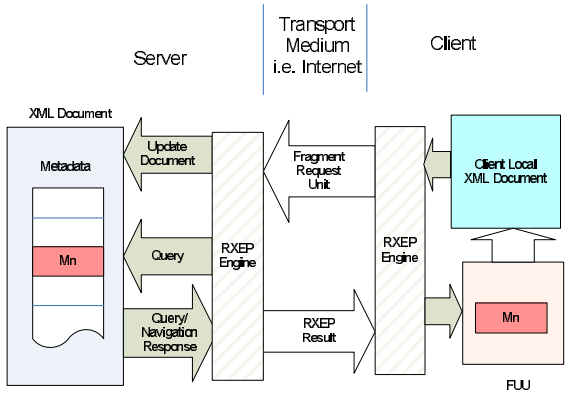

Fig. 4. Interaction of FRUs and FUUs

- Update - updates the fragment specified by the XPath locator; and
- Insert – Inserts a node before the specified XPath locator.

RXEP FUUs are implemented in XML Schema and thus differ from the FUUs that appear in MPEG-B, as they provide the additional flexibility of allowing 3rd party applications to extend the schema. An example of an RXEP FUU, as a response to the RXEP FRU in Fig. 2, is illustrated in Fig. 3. This FUU demonstrates addition of the XML under the *ADD* node to the location specified by the XPath locator, in this case */Media/Music/Song[1]*

C. *Overall System*

This section considers the overall combination of RXEP FRUs and RXEP and is illustrated in Fig. 4. Clients first initiate a connection to the sending peer and request an XML document; the received XML is then a 'partial skeleton' of the complete XML document. The client can then construct extra FRUs to request further fragments of the remote XML document. These FRUs may simply contain navigation commands, or in more 'intelligent' cases, XPath queries to retrieve multiple fragments. The server peer decodes the FRUs and generates fragments customised to the individual client. The following examples refer to the XML shown in Fig. 1.

1) *Navigation Example*

Imagine a portable device with a small screen (which displays 10 lines of text) and limited memory. A FRU requests an XML Document, and receives the root node, *Media.* The User issues a RXEP XMLPull Expand command, which returns the first child, *Music*. The user issues another expand and receives the first song node. This is not the node required and the user issues an XMLPull Next command, receiving the next sibling. The user decides this is the node and expands the node, followed by continued navigation or, perhaps, requests for that entire branch.

2) *Query Example*

A user connects to a remote server and requests an XML document. The user wishes to select all songs from the Media collection. The user thus creates an FRU with the XPath expression (i.e. \\Song to request all Song nodes), and receives an FUU with multiple ADD commands each con-

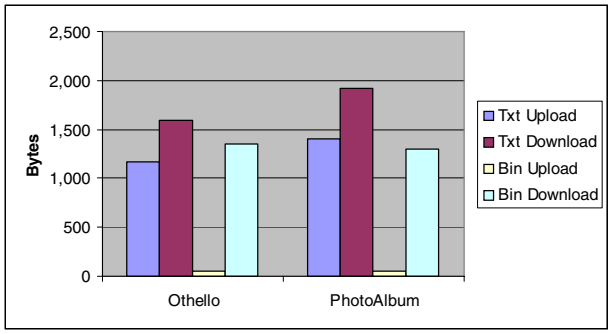

Fig 5. Comparison of upload and download of tests

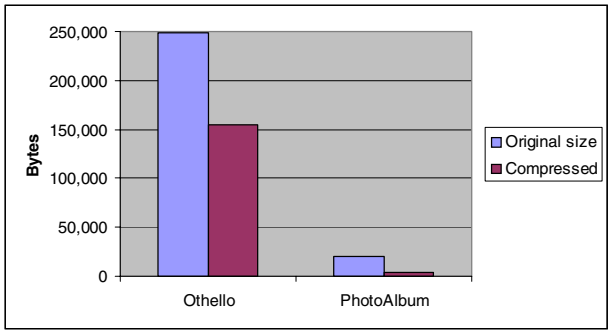

Fig 6. Comparison of text and binary compression

taining a fragment in response to the query. Thus, from one, simple FRU the user has received all information without retrieving the whole XML document.

D. *BinRXEP*

Binary RXEP (BinRXEP) [7] is the compressed binary representation of the RXEP XML which applies to both the FRUs and FUUs. Compression exploits the tree-based structure of the XML as well as a priori knowledge from the XML schemas. Although the compression technique is highly dependent on the structure of the Schemas, significant savings arise from elimination of the need to encode the full XML tag names. For example, a node in the Schema defines a choice of four children, thus, only two bits are required to represent a choice of any one child.

One advantage of this method of compression is that the binary file still retains the same structure as the original text XML. This allows sections of a binary XML file to be added or read, without the need to decompress the entire file (which is the case with non-XML aware character redundancy compression techniques).

Through extension of the RXEP Schemas (i.e. adding extra nodes to the original schema), additional information, vital to the Binarisation process of the XML document, can be delivered to the client. This includes the Namespace information (to allow a client to retrieve and load the correct Schema used in the compression), and most importantly, the code indicating the global element of the original XML document. This global element code allows the client to select the root XML node to commence decompression.

III. Collaborative Editing

Collaborative editing is a process that occurs on a day-to-day basis and increasingly there are multiple authors editing a single document. One example is the popularity of 'Wikis' [8], which have become accepted for collaborative editing on the Internet. Applications of Wikis range from software documentation/howtos to online encyclopedias (such as Wikipedia). Programmers also regularly collaboratively edit software code using versioning software such as CVS [9]. This allows an author to check for updates, compare against their local version before making modifications, to ensure they have the latest version of file.

Office document formats, such as OASIS OpenDocument and Microsoft Office documents have begun using XML as the container to store data. It is thus possible to utilize standard XML tools and collaborative editing techniques on these office documents.

RXEP becomes an ideal candidate for managing the delivery and request of XML documents. This is beneficial since often the entire document is not desired. Additionally, features of RXEP such as add, delete, update and insert provide collaborative editing functionality.

Using RXEP, a client may download parts (or all) of a document, make changes locally, and only upload the parts that have been changed (upload entire fragment or just a diff). For example, a User who wishes to revise a document can create FRUs and navigate through the document. On finding an error in the document they can make changes and only those changed fragments of XML need be relayed back to the server as an FUU to update the original document.

RXEP also provides additional functionality for users on portable devices. For example, users may only wish to retrieve one paragraph at a time to reduce storage requirements, bandwidth usage, or download waiting times. RXEP can thus be used to make browsing of large documents convenient and practical.

Strict collaborative editing rules may utilize the tree based structure of the XML. Branches of the tree can be locked (so no other authors can edit), whilst allowing the other branches to be edited; this allows separate portions of the document to be edited at the same time, while preventing others updating the sections that are being worked on.

IV. Results

To demonstrate the effectiveness of RXEP for collaborative editing and delivery, we consider some practical examples. These were evaluated using an implementation of RXEP and BinRXEP in JAVA. These examples are aimed at investigating the amount of data transferred in the process of viewing/editing particular documents. For the Binarisation, FUUs are used to indicate to the client which Schema and associated Namespaces are used for Binarisation of the XML document.

A. *Receive XML Documents with RXEP*

The first set of results examines RXEP's efficiency when downloading only sections of data. Here textual and binary

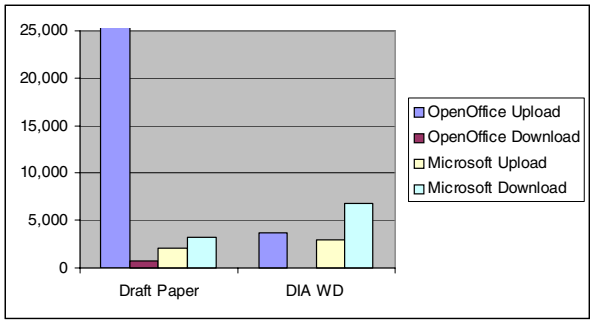

Fig. 7. RXEP using Openoffice and Microsoft

representations are compared to the original document size and results are taken from the upload and download traffic totals from the client's device.

The first text file for comparison is an XML version of Shakespeare's Othello. This file contains little structure and mostly string text, with an original file size of 248,777 bytes. The test was performed by navigating to Act 8->Scene1->1st speaker and changing the speaker name, and uploading the alteration. The text mode resulted in a 1,176 bytes upload and 1,598 bytes download while the binary mode resulted in a 50 bytes upload and 1,349 bytes download. To Stream the text to that position (i.e. downloading the whole file to that point) would have required 207,030 bytes; this demonstrates the savings by skipping the unwanted parts of a file.

The second example consists of a photo album represented by a MPEG-21 Digital Item. Initially, this XML file is of length 21,040 bytes. The test navigated to the level of photo descriptors and altered the description of a photo. The text mode resulted in a 1,409 bytes upload and 1,914 bytes download while the binary method results in a 54 bytes upload and 1,307 bytes download. To stream to the chosen location in text would require 3,997 bytes.

Fig. 5 graphically illustrates the comparison of RXEP and BinRXEP with the simple examples while Fig. 6. illustrates the comparison of text and binary representation of the complete test files. Since these are very simple examples, and navigation skips most of the unnecessary structure information, the text and binary download results are not significant. However, more complex examples, involving more navigation through the structure would see further improvements; such as the savings seen in Fig. 6. The results show that using BinRXEP the uploaded data is very small and is a minimal cost for the flexibility afforded by RXEP.

B. Comparison of Office XML Documents with RXEP

This section will examine how the office XML documents, OpenOffice and MS Word, compare when used in conjunction with RXEP. These results are shown in Fig. 7.

The first test document (DIA WD) consists of a MPEG-21 Digital Item Adaptation working draft document. The document was converted into xml using both software packages. The test consisted of navigate to the third section, then navigate to a paragraph, where as mistake is present. The new fragment containing the correction was then uploaded. OpenOffice XML required 27,742 bytes download and 836 bytes upload. The MS Word XML required 3,282 bytes download and 2,108 bytes upload.

```
<w:wordDocument ……..  >
  ….
  <w:body>
    <wx:sect>
      <w:p>
        <w:r>
            <w:t>This is a Test sentence.</w:t>
        </w:r>
      </w:p>
      <w:p/>
      <w:p/>
      <w:sectPr>
        <w:pgSz w:w="12240" w:h="15840"/>
        <w:pgMar w:top="1440" w:right="1800"
            w:bottom="1440" w:left="1800"
             w:header="708" w:footer="708"
            w:gutter="0"/>
        <w:cols w:space="708"/>
        <w:docGrid w:line-pitch="360"/>
      </w:sectPr>
    </wx:sect>
  </w:body>
</w:wordDocument>
```

Fig. 8. Sample of a Wordprocessing ML document

The second file (Draft Paper) consisted of the first draft of this paper, converted using both packages. The test involved navigating to the RXEP section and modifying the second paragraph. OpenOffice XML required 7,529 bytes download and 3,723 bytes upload. The Microsoft (MS) Word XML required 6,787 bytes download and 2,910 bytes uploaded.

Interestingly, it was found that in the first example, the OpenOffice XML used a much flatter document structure, where many elements occurred at the body node level, requiring significantly more download as compared to the MS Word XML. In the second example, the file had fewer major sections than the first file, and the differences were not as significant. Furthermore, the OpenOffice XML was much harder to navigate due to the flatter structure than the MS Word XML. Overall, this demonstrates that document structure can adversely affect RXEP's efficiency.

It should be noted that binarisation was not tested for the OpenDocument since the Schemas are written in Relax-NG, which is not supported by the current version of the compression code. Additionally, the MS WordprocessingML documents could not take full advantage of the tree-based compression, as discussed in the next Section.

C. Discussion

It was found that the Microsoft Office XML documents cannot be fully compressed using tree-based techniques. The problem arises in our example after the *wx:sect* node. Fig. 8. illustrates a portion of the WordprocessingML test document concentrating on the body of the document. The corresponding Schema for this node is shown in Fig. 9. As can be seen in Fig. 9, the *sectElt* type (which is the type of the sect node) has a sequence of a *sub-section* or an *any* node. Additionally, the *sub-section* type also contains an *any* node in its sequence.

```
<xsd:element name="sect" type="sectElt">
</xsd:element>

<xsd:complexType name="sectElt">
      <xsd:sequence>
        <xsd:element name="sub-section"
                     type="subsectionElt"
                  minOccurs="0" maxOc-
curs="unbounded">
        </xsd:element>
        <xsd:any namespace="##other"
             processContents="lax" minOccurs="0"
             maxOccurs="unbounded"/>
      </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="subsectionElt">
      <xsd:sequence>
        <xsd:any namespace="##other"
            processContents="lax" minOccurs="0"
             maxOccurs="unbounded"/>
      </xsd:sequence>
</xsd:complexType>
```

Fig. 9. Relevant portion of the Wordprocessing ML Schemas

The *any* node in this case uses the "lax" processing directive which allows any "well-formed" XML to be present, and validation is done on a "can do" basis [2]. Whilst the *any* node allows for flexibility, this has the unfortunate expense of limiting its compressibility. The problem is that tree-based encoding relies on knowledge of the possible nodes that can be present after the current node. Unfortunately, the compressor cannot easily determine what node must come next after an *any* node with the lax processing directive.

One solution is to restrict nodes following an *any* node to be global elements from a known namespace. In terms of binary, one bit would be used to indicate if the node is valid to a Namespace and the following bits to select the Namespace and the selected Node within that Namespace. Without this, it is not possible to determine the next node, in relation to the schema, and to continue compression using the Schema tree-based technique. If the schema node cannot be determined, then the contents of all XML under that node will be compressed with a standard character redundancy compression technique, such as zlib [10]. This approach is less flexible and achieves reduced compression efficiency as compared to the tree-based compression [11, 12]. One further disadvantage is the inability to directly 'edit' small portions the binary data, when using zlib, while the schema binarisation approach retains that possibility. For example, if the document is stored in binary, then to simply insert some data into the document (under the *any* node branch), the entire branch needs to be decompressed before the addition can be made, then recompression is required. Where tree-based compression is used, the new data can easily be inserted, without the need to decompress the entire branch.

In this example (see Fig. 9.), since the *Section* element is an *any* node and the following paragraph *p* element is not a global element of the corresponding Namespace, thus efficient, schema based compression is not feasible.

V. Conclusion

The common office suites are moving towards XML documents for file storage and this emphasizes the need for a standard approach to XML collaborative editing. This paper has demonstrated that RXEP is a good candidate for receiving and editing remote XML documents. Using the binary, compressed for of the protocol, we have shown that upstream data transmissions are negligible and the advantages of minimizing downstream data transmissions far outweigh any disadvantages of a small upstream data transmission. The paper has also highlighted that some of the current office suites do not produce ideal XML in term of compression and that there is still a lack of uniformity in the choice of schema standards. Given that there is progress on these issues, it is expected that XML tools such as RXEP will provide versatile mechanisms for standard XML-based office suite collaboration.

References

[1] W3C, "Extensible Markup Language (XML)," [online document] http://www.w3.org/XML/.

[2] W3C, [online document] "XML Schema," http://www.w3.org/XML/Schema

[3] OASIS, "Open Document Format for Office Applications (OpenDocument)", [online document] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=office

[4] Microsoft, "Office 2003 XML Reference Schemas", [online document] http://www.microsoft.com/office/xml/default.mspx

[5] S. Boettcher and A. Turling, "XML Fragment Caching for Small Mobile Internet Devices," *Web, Web-Services, and Database Systems, NODe 2002 Web and Database-Related Workshops, Erfurt, Germany*, pp. 268–279, October 7-10 2002.

[6] Information technology, "MPEG B -- Part 1: Binary MPEG format for XML," ISO/IEC 23001-1:2005.

[7] S. Davis and I. Burnett, "Exchanging XML Multimedia Containers using A Binary XML Protocol," *IEEE ICME 05*, 2005

[8] "Wikipedia", http://en.wikipedia.org/wiki/Main_Page

[9] "Concurrent Versions System", [online documen] https://www.cvshome.org

[10] "zlib", [online document] http://www.gzip.org.zlib

[11] S. Davis and I. Burnett, "Efficient Delivery in the MPEG-21 Framework", to be published, *1st International Conference on Automated Production of Cross Media Content for Multi-channel Distribution (AXMEDIS)*

[12] M. Cokus and D. Winkowski, "XML Sizing and Compression Study For Military Wireless Data," *XML Conference and Exposition 2002*, Baltimore convention center, Baltimore, MS USA, December 8-13 2002.