EXCHANGING XML MULTIMEDIA CONTAINERS USING A BINARY XML PROTOCOL

Stephen J. Davis and Ian S. Burnett

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

ABSTRACT

XML is becoming increasingly popular as the ubiquitous standard for metadata; consequently, it is being incorporated into many multimedia applications, such as those based on MPEG-7 and MPEG-21. However, XML is often verbose and transmitting the large files can be wasteful in bandwidth and power-limited mobile applications. This paper introduces an XML access mechanism, RXEP, which combines XML compression with a fragment access protocol. RXEP ensures that essential information is exchanged efficiently while minimising superfluous XML content transmission. This makes XML containers an attractive technique for multimedia content delivery.

1. INTRODUCTION

XML [1] and XML schema [2] have achieved significant acceptance for data exchange due to their inherent structure and human readability. However, the advantages of a textual format (the native format of the WWW) are also major disadvantages for efficient communication in e.g bandwidth limited mobile environments. Recently, compression techniques [3–6] have alleviated these issues and thus XML schema becomes an attractive mechanism for the packaging of multimedia content for delivery in all environments rather than just on the internet. Several standards have built 'content containers' using XML but the most complete approach is provided in the MPEG-21 Digital Item [7]. Using XML as the 'container' for multimedia content makes the significant set of existing XML tools available for multimedia delivery. Examples of this include the several XML parsing techniques for different situations such as DOM and SAX, which allow XML to be parsed on a wide range of devices. Most importantly, however, XML multimedia containers can facilitate interoperatibility amongst devices, by incorporating metadata describing content and how it could/should be used.

While the interoperatibility afforded by XML containers has clear advantages, a significant drawback in mobile environments is the large volume of XML descriptors which may need to be incorporated to cater for diverse scenarios, content and applications. While, these can be compressed to some degree, they inevitably waste bandwidth, processing power and limited mobile device memory. A solution to these problems would be to only deliver the XML fragments [8] which are relevant (i.e. keeping the XML primarily on a server and restricting delivery to a mobile client). Some of these issues are addressed in [9] for text based XML, but in this paper we significantly expand that work to create a full two-way protocol for the delivery of XML fragments. Remote XML Exchange Protocol (RXEP) is fully defined in XML Schema but may be compressed using schema based compression to minimise bandwidth costs. Thus, RXEP offers all the structural and processing advantages of XML/XML schema while minimising costs in the mobile environment. It achieves this by explicitly utilising the two-way communication available on networks.

This work was partially funded by the Smart Internet Technology CRC

```
<RXEP xmlns="RXEP:2004">
   <Add location="/Media/Music/Song[2]"
        ns="mediaNS:2004">
      <Title>Hit.2</Title>
      <Description>Song 2</Description>
      <Artist>B. Artist</Artist>
      <Format>OGG</Format>
      <Length>03:46</Length>
   </Add>
</RXEP>
```

Fig. 1. Example RXEP packet

The most commonly used XML protocol is Simple Object Access Protocol (SOAP) [10] this is a text based format. This is a disadvantage for protocol-based exchange of binary content embedded in XML files as such data must be converted to base64 [11] increasing the data size by 33%. There have been binary encapsulation proposals such as DIME [12] but this ignores the XML structure in favour of a new binary format. Alternatively, 'SOAP attachments' and XML-binary Optimised Packaging [13] avoid the problem by referencing binary content rather than conveying it in the message. RXEP is, instead, an XML defined protocol which can be binarised; this means that binary content can be conveyed natively in the binary format of RXEP while protocol messages are still entirely XML text compliant when in RXEP text form. This avoids conversions to base64 thus saving bits and keeps to the 'text-only' mantra of the XML. While RXEP messages could be encapsulated in SOAP messages this appears to be a needless overhead for what is intended to be a bit-efficient protocol.

In the following sections, we overview RXEP, summarise XML binarisation/compression and then, detail the mechanisms by which binarisation is applied to RXEP commands and the XML fragment responses.

2. REMOTE XML EXCHANGE PROTOCOL

Remote XML Exchange Protocol (RXEP) is a new method for requesting relevant data from an XML document located on a remote server (or peer). RXEP has the ability to create XML fragments building upon pull-parsing techniques [14] (navigating on an element-by-element basis), or from queries (i.e. using XPath). RXEP queries can receive multiple fragments at once, i.e. XPath query //Item, to retrieve all Items in the remote XML document. RXEP can operate on a request-response basis or in streaming mode. In the request-response scenario, commands are used to send instructions to the server. A brief summary of RXEP com-

Table 1. RXEP Commands

Command	Definition
Get	Initiates a connection and identifies an XML file to be exchanged.
XPath	Specifies an XPath expression requesting an XML Element(s).
XML-Pull commands (client to server):	
Next	Retrieves the next sibling node in the order that the nodes appears in the XML document.
Expand	Expands the current node, allowing access to the children.
Up	Moves up to the parent node.
Back	Moves to the previous sibling.
RXEP commands:	
Add	Adds the fragment to the client's local XML instance.
Delete	Deletes the specified fragment from the client's local XML instance.
Update	Updates the fragment in the client's local XML instance.
Insert	Inserts the fragment before/after the specified element in the clients local XML instance.

mands are given in Table 1 and further information can be found at [15]. XML fragments generated from queries are packaged into the RXEP XML and transmitted to the client. In this paper, we will only consider in detail the RXEP Add command, which instructs the client to add the XML fragment to its local version, using a location attribute to specify the XPath location. A simple example of an Add command for RXEP is illustrated in Fig. 1, the fragment is derived from the example XML Document as shown in Fig. 3. The remaining RXEP commands such as, delete, insert and update, provide for full collaborative editing between peers. This illustrates the versatility and wider application of RXEP.

3. SUMMARY OF XML BINARISATION

Generally compression of XML files is referred to as binarisation i.e. the conversion from human-readable textual format to a binary format. To date, a number of XML compression techniques have been reported; typically they can be classified as redundancy, schema or hybrid (a combination of the redundancy and schema) methods. The most notable methods are, Millau [3], xmill [4], MPEG-7 Binary format for MPEG-7 Metadata (BiM) [5] and xmlppm [6]. Studies of XML compression techniques, such as [16], as well as our own tests, reveal that, on average, tree/schema based schemes such as BiM achieve the lowest data filesize.

Some techniques, such as BiM [5], support one-way streaming, which is particularly useful when streaming multimedia, and receiving associated metadata relevant to the media (e.g lyrics with the associated part of a music file). A shortcoming of all the compression techniques, is that there is no mechanism to allow the client to request and retrieve just the information required, or to simply navigate though an XML document and receive responses in binary. These are both important facets of systems delivering complex multimedia containers and combining binarisation with RXEP offers a suitable solution.

4. BINARISATION IN RXEP (BINRXEP)

Native RXEP commands are expressed in XML valid to an associated XML Schema; the protocol itself is thus inherently able to

```
Media
  [CHOICE] {1,Unbounded}
   Music (0) {0,1}
     [CHOICE] {0,Unbounded}
        Song (0) {0,1}
          [SEQUENCE] {1,1}
            Title        {1,1}
            Description  {0,1}
            Artist       {1,1}
            Format       {1,0}
            Rating       {0,1}
            Length       {0,1}
      ...
   Videos (1)
       ...
```

Fig. 2. Tree View of Schema

be binarised. Thus, the problem remaining is the binarisation of randomly requested fragments of the XML file. While any compression technique can be applied to the fragments, this is most efficiently performed when the XML is schema valid and schema based compression (e.g. BiM) can be employed. BiM is not able to perform this task directly because it does not possess mechanisms to control the binarisation process - an assumption is made that the whole XML document will eventually be transmitted. Generally, BiM and schema-based compression schemes will encode the entire document subtree at once, but provides one exception, deferred nodes; that mechanism is, however, not helpful when the requirement is to 'navigate' remotely through the XML document. Thus, BinRXEP extends tree/schema compression techniques to incorporate controls on the binarisation derived from the RXEP commands (i.e. node location, level depth). In particular, this requires that the binarisation is applied only to direct children of nodes with all their information such as attributes and values. It is also possible to specify level depths to request all descendants of a node to a certain level and, further, to utilise full XPath expressions to access multiple elements and descendants in a single request. Thus RXEP provides a complete binary/compressed protocol specifying versatile requests and delivery formats for randomly accessed fragments.

BinRXEP is used for the protocol commands and optionally for XML fragments (when they are schema valid). Essentially, BinRXEP is a schema/tree based compression scheme which relies on the fact that the metadata structure (the XML Schema) is known by both the client and sender. Thus, each schema element can be assigned a unique binary code. This eliminates the need to send the entire element tag as a string, which results in significant bit savings. For demonstration purposes, we have represented a portion of an example schema, illustrated in Fig. 2, in a tree view. The generated binary codes are surrounded by round brackets, i.e. (010) defines the second child. The minimum occurrence and maximum occurrence of all nodes are surrounded by curly braces respectively, i.e. {0, Unbounded} indicates that the node does not need to occur, and there is no upper bound on the number of times it may appear. To illustrate the bit savings, we will refer to the Media element from the schema in Fig. 2. Here, the Media element has a choice of two children, which are Music and Videos. Since there are only two options, this can be represented with just one bit. Thus, the Music node, which is 40 bits as a string, can be represented by its binary code of just 1 bit, '0'.

Beyond the basic BinRXEP schema/tree mechanisms, it has been necessary to introduce specific mechanisms to cater for nav-

```
<Media xmlns="mediaNS:2004">
   <Music>
     <Song id="Hit1">
       <Title>Hit.1</Title>
       <Description>Song 1</Description>
       <Artist>A. Artist</Artist>
       <Format>MP3</Format>
       <Length>02:23</Length>
     </Song>
     <Song id="Hit2">
       <Title>Hit.2</Title>
       <Description>Song 2</Description>
       <Artist>B. Artist</Artist>
       <Format>OGG</Format>
       <Length>03:46</Length>
     </Song>
   </Music>
</Media>
```

Fig. 3. Example XML

igation and binarisation of XPath expressions. These are detailed in the following subsections.

4.1. Binary XPath Locators in BinRXEP

XPath locators, within RXEP, can be used to specify the element for retrieval. An XPath locator is defined as the XPath expression which specifies the path and position of an element in an XML file. These locators are used with the RXEP response commands such as Add, Delete, Update and Insert. Since the XML is valid to a schema, then we know that the XPath Locators must follow the schema rules to be valid. Exploiting this information, we can apply the same binarisation techniques as we do to the XML. This process is slightly different however, as the Binarised XPath Locators do not contain the XML model group information and the action is always a choice (identifying exactly one node). The binarisation of an XPath locator has three steps: 1. get the binary code for current element; 2. if the parent allows multiple children, then a mandatory integer needs to be encoded to indicate the position of the child element; and 3. an integer indicating the child element counter (for multiply occurring child nodes).

For example, in Fig. 1 the XPath locator is "/Media/Music/Song[2]". Since Media is the only root node, it is mandatory and thus no bits are required. The binary code for Music is 0 and since there can be an unbounded number of Music (via the choice), a position code of one is encoded using VLC5, i.e. 00001. Following the same process for Song[2] we get a 0 to select the Song node and 00010 for the second position code. Since there may be a number of Songs the 'counter' is needed to represent the [2], which would be 00010 as VLC5. The total output is 00000100001000010 (17 bits) which is significantly less then the string representation of 160 bits.

This binary representation contains an additional number indicating where element is positioned (see step 2 above). This allows us to exactly place the element in its correct position preserving element order (if necessary). For example, if the Music node had 3 children, Song, Other and Song, /Media/Music/Song[2] from the XPath selects the second song (third child), however, at the client side, if nodes are not retrieved in order, i.e from a query, then its position of three is unknown from the XPath.

4.2. Navigation with BinRXEP encoded XPath Locators

Since we have an efficient method of representing the XPath locators as binary, which are used for fragment location and position. The navigation part of the protocol will ensure that for each element selected by the XPath locator, all direct child nodes are returned. Referring to Fig. 2 and Fig. 3, we will illustrate this process via a simple example. Initially, after an initial RXEP GET request, the document root node, Media, is selected (server side). The namespace URIs are transmitted to the client, in order, along with the root node binary code. This now gives the client enough information to load all necessary schemas and create binary codes for the decompression process. The modelgroup [CHOICE] and only its direct child nodes are then encoded. First, using VLC5 which is a method for encoding numbers of an unknown size [5], the number of choices are encoded; in the example shown in Fig. 3 there are 2 children so 00001 is written. Here we see that the XML element Music is present in this XML instance, thus a 0 is written which is the choice code to select Music. Since this node does not allow attributes we continue to the next sibling. The second choice is Videos, and a 1 is written. The total bits sent to the client in this request would be just seven bits, i.e. 0000101. This process would continue for the next requested node.

4.3. Navigation with BinRXEP XML-Pull

We will now consider using XML-Pull commands (refer to Table 1) rather than XPath Locators. Instead of receiving all the direct child nodes from a selected node, here we step though the XML on a per-node basis. This is analogous to XML Remote Pull-parsing but by using RXEP the 'parsing' is truly remote (i.e. across a network).

The same example as in Section 4.2 will be used and the binary requests and responses for the XML-Pull commands are given in Table 2. The XML document is requested and the code representing the root element is returned, i.e. Media. The client sends the binary code representing the *Expand* command, which instructs the server to expand the Media node and returns the binary code for the first child node, in this case '0', the code for Music. The user selects music which sends the binary code *expand*, the received binary code '0' indicates that a Song element is present and the following bits represent the attribute 'Hit1'. The user does not want this song and the binary code for *next* is sent which moves the position to the next sibling on the server. The server sends the binary bit '0' to indicate another Song element is present and the following bits indicate the attribute 'Hit2'. The user determines this is the desired song, and sends the binary code for *Expand*, in this case the first element Title is mandatory and no code is needed to represent the element. The bits received indicate the value for the Title of the requested song.

4.4. BinRXEP Fragments from Queries

There are many cases when it is desirable to request multiple elements or fragments in a single query and this is facilitated using XPath queries in RXEP. The process will be illustrated using the simple 'Example XML' in Fig. 3 and the RXEP packet shown in Fig. 1. For the example, the XPath for the query is assumed to be //Song[@id="Hit2"] which asks for all songs beginning from the root node, which also have an attribute ID "Hit2". In the XML of Fig. 3, there is only one matching element and the fragment of XML, returned by RXEP, will be that element and its immediate children; this fragment is placed between the Add tags of RXEP. Fig. 1 shows the complete RXEP packet response for this query and illustrates the extra information which must be provided

Table 2. RXEP XML-Pull Navigation

Request	Binary Response	Description
Expand	0	Expands the Media Node and returns its 1st child, Music
Expand	0 plus (bits for attribute id value “Hit1”)	This indicates the Song Element and its id attribute
NEXT	0 plus (bits for attribute id value “Hit2”)	This indicates the 2nd Song element and its id value
Expand	(bits for Title value “Hit.1”)	Expands the Song node and returns the 1st child, Title

to the client alongside the fragment itself. This is provided using an XPath locator and its corresponding namespace; in this case, /Media/Music/Song[2] and *ns*. This Locator tells the client the exact starting location within the Schema to begin decompression of this fragment, while the *ns* attribute selects the correct namespace to evaluate the XPath locator. For binarisation, the binary codes will follow the same procedure as given in the example in Section 4.2. In more complex cases, where a level depth parameter is included with the query, the encoder continues down all subtrees until reaching either the end of the subtree, or the desired level. When nodes are selected on the basis of a Query (from the client), it cannot be assumed that the client already has the element, thus it must be sent, along with its information such as the ‘id’ attribute in the example. The resulting binary bitstream for the RXEP packet in Fig. 1 would be:

1. The RXEP node is mandatory and no code is needed
2. Two bits to represent the Add command e.g. (00)
3. One bit (1) to represent the non-mandatory Location attribute and bits to represent the value “/Media/Music/Song[2]” encoded as a binary XPath locator e.g. (00000100001000010)
4. One bit (1) to represent *ns* attribute and bits to represent the value “mediaNS:2004” encoded as a String
5. Bits for encoding the value “Hit2” (Song id=’Hit2’)
6. Sequence is mandatory - no code is needed
7. One bit (1) - the Title tag is present in the XML
8. Bits to encode value “Hit.2” as a String
9. One bit (1) - Description tag is present in the XML
10. Bits to encode value “Song 2” as a String
11. One bit (1) - that the Artist tag is present in the XML
12. Bits to encode value “B. Artist” as a String
13. One bit (1) - Format tag is present in the XML
14. Bits to encode value “OGG” as a String type
15. One bit (0) - Rating tag is not used in the XML
16. One bit (1) - Length Tag is present in the XML
17. Bits to encode value “03:46” as a Time type

4.5. Embedded Resources with BinRXEP

BinRXEP has the ability to retrieve desired fragments from an XML document, this also applies to embedded resources within the XML. The ability to embed resources within XML has it advantages, most notably, eliminating the need to send a URL pointing to a resource and the client retrieving itself, often using additional protocols. However, currently embedding resources into XML has the penalty of increased resource size (due to base64 encoding) as well as overall file size of the XML (many embedded base64 encoded resources). Using BinRXEP, these factors are no longer an issue since everything is transmitted in binary. Thus for BinRXEP, the base64 size increase is eliminated and most importantly, only selected parts of the XML are retrieved, i.e. unused embedded resources are not retrieved.

5. CONCLUSION

This paper has demonstrated the effectiveness of combining RXEP with a binarisation technique (tree/schema compression) to efficiently transmit required parts of an XML document. Access to the required parts of the XML can be achieved by remotely navigating or querying the XML file. By eliminating unnecessary data transmission, RXEP significantly reduces overall bandwidth and storage requirements. The extension of RXEP to XML files with embedded resources highlights the potential of compressed XML as a multimedia container format.

6. REFERENCES

[1] W3C, “Extensible Markup Language (XML) 1.0 (Third Edition),” `http://www.w3.org/TR/REC-xml/`, 04 February 2004.

[2] W3C, “XML Schema Part 0: Primer,” `http://www.w3.org/TR/xmlschema-0/`, 2 May 2001.

[3] M. Girardot and N. Sundaresan, “Efficient Representation and streaming of XML content over the Internet medium,” *IEEE ICME 2000*, vol. 1, 2000.

[4] H. Liefke and D. Suciu, “XMill: An Efficient Compressor for XML Data,” *Technical Report MS-CIS-99-26, Univ. of Pennsylvania*, 1999.

[5] MPEG, “ISO/IEC 15938-1:2001, Information Technology - Multimedia Content Description Interface - Part 1: Systems,” 2001.

[6] J. Cheney, “Compressing XML with Multiplexed Hierarchical PPM Models,” `http://www.cs.cornell.edu/People/jcheney/xmlppm/paper/paper.html`, 2000.

[7] J. Bormans and K. Hill, “MPEG-21 Overview v.5,” *ISO/IEC JTC1/SC29/WG11/N5231*, October 2002.

[8] W3C, “XML Fragment Interchange,” `http://www.w3.org/TR/xml-fragment`, 2001.

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

[10] W3C, “SOAP,” `http://www.w3.org/TR/soap/`.

[11] RFC, “Multipurpose Internet Mail Extensions Pt.1, rfc-2045,” 1996.

[12] Microsoft, “Direct Internet Message Encapsulation (DIME),” `http://msdn.microsoft.com`.

[13] W3C, “XML-binary Optimized Packaging,” `http://www.w3.org/TR/xop10/`, 16th November 2004.

[14] XMLPULL, “XML Pull Parsing Common API,” `http://www.xmlpull.org`.

[15] S. Davis and I. Burnett, “Remote XML Exchange Protocol,” `http://www.whisper.elec.uow.edu.au/people/sdavis/RXEP.html`, 2005.

[16] M. Cokus and D. Winkowski, “XML Sizing and Compression Study For Military Wireless Data,” *XML Conference and Exposition 2002, Baltimore convention center, Blatimore, MD USA*, Dec. 2002.