QUERY STREAMING FOR MULTIMEDIA QUERY BY CONTENT FROM MOBILE DEVICES

Kevin Adistambha, Stephen J. Davis, Christian H. Ritz and Ian S. Burnett

Whisper Labs, School of Electrical and Telecommunications Engineering,
University of Wollongong, Australia

ABSTRACT

Formulating and processing of multimedia queries using mobile devices presents many challenges. This is due to the limitations of the devices themselves and the cost of the bandwidth involved in transmitting multimedia data between servers and devices. In this paper we propose a novel approach: “query streaming” which uses Reverse Polish Notation to perform multimedia query-by-example on a mobile device and server. An important advantage of query streaming is the ability to perform a query within the previous result set. To solve the problem of limited resources, the concept of result set examination using Fragment Request Units and Fragment Update Units is also explored. MPEG BiM compression is performed on the communication messages to further minimize transmission requirements.

1. INTRODUCTION

The area of multimedia query has received considerable attention lately, as a result of the phenomenal growth of user created multimedia content. This explosion in user generated content is visible in the popularity of multimedia sharing sites such as YouTube [1], VideoJug [2] and Flickr [3]. Predictably, searching those multimedia items is a non-trivial task that requires metadata to be attached to the multimedia items in question. Examples are the low-level descriptions of MPEG-7 [4] and higher-level, semantic metadata such as Dublin Core [5]. Attaching the correct description to the multimedia items remains a major problem to be solved today, due to the volume of source data to be described. Requiring a human to sift through all the data he/she created is non-optimal due to the intensive nature of the task and subjectivity involved. Some automatic and semi-automatic description generation have been developed (such as IBM’s MARVEL [6]) but these remains experimental.

While description generation has been partially addressed, one basic problem remains: a standard communication language between clients and database solutions such as MARVEL, Google Image and others. Some research has been done in this area, notably MRML [7], MOQL [8] and the author’s Multimedia Query Format (MQF) [9]. However, these approaches did not specifically consider the restrictions imposed by mobile devices. In this paper, we investigate the use of a mobile device to perform multimedia query (as depicted in Fig. 1).

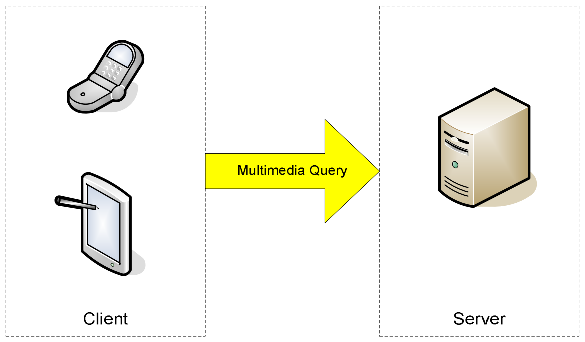

Figure 1. Multimedia query using mobile devices.

In today’s increasingly mobile environment, it is inevitable that multimedia queries will be performed on mobile devices. Compared to desktop PCs, such devices have specific limitations such as small screen size, limited memory and processing power and high bandwidth cost. The hardware limitations of mobile devices are outside the scope of this paper; here the focus is exclusively on bandwidth usage and the cost associated.

Specifically, this paper addresses two areas:

- Bandwidth: Performing multimedia queries is obviously a bandwidth-intensive operation; hence some measures must be taken to minimize data transmitted from the mobile devices to the server because of the cost involved. Therefore, any reduction in bandwidth usage would be helpful in this case.
- Physical limitations: Size limitations of mobile devices makes typing inconvenient, at best. A method to refine a multimedia query based on an existing query without having to perform another query is therefore desirable. Hence, the aim would be to search within a previous query result set with minimal effort and overhead.

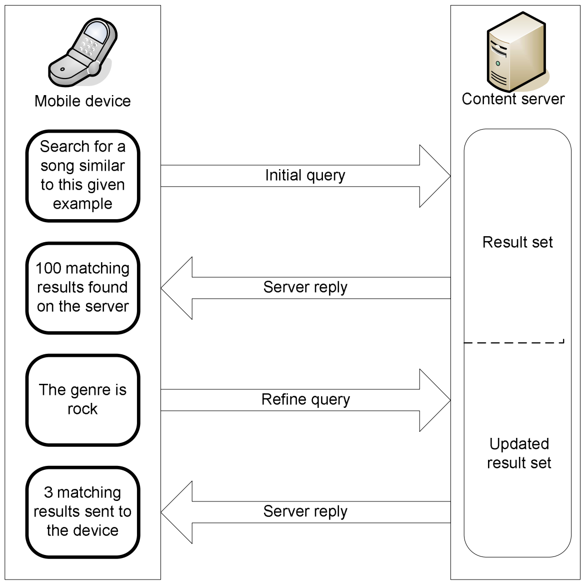
Mobile device
Content server
Search for a song similar to this given example
Initial query
Result set
100 matching results found on the server
Server reply
The genre is rock
Refine query
Updated result set
3 matching results sent to the device
Server reply

Figure 2. Application scenario using query streaming.

One of the fundamental difficulties with query streaming is the need to progressively construct the query and corresponding responses. In [9], the authors proposed MQF as a Reverse Polish Notation (RPN) based query format. In Section 2 of this paper we discuss RPN and its application to query streaming before briefly overviewing MQF. The foregoing Sections then consider combining MQF query streaming with fragment approaches to XML retrieval. This results in efficient querying and results set treatment in mobile scenarios.

2. QUERY STREAMING: APPLICATIONS, RPN AND MQF

2.1 QUERY STREAMING APPLICATIONS

Two potential application scenarios using query streaming are now considered:

1. This scenario is illustrated in Fig. 2. Using a mobile device, a user would like to search for a particular song but he/she only has a portion of the song. The server has 100 similar matches to the song in question, and the user would like to refine the search while avoiding the need to listen to each potential match due to the bandwidth limitations of mobile devices. The user might also further refine the query by sending additional restrictions, thus reducing the number of potential matches to a manageable number (preferably just one).
2. In a second scenario, a user is lost (or wishes to identify e.g. a building) in an unfamiliar city. The user takes pictures of the surrounding environment and sends a query-by-example to a server. The server returns a message saying that the image sent by the user is too general and it has 50 sites that look similar to the picture. The client then takes another picture to further refine the search.

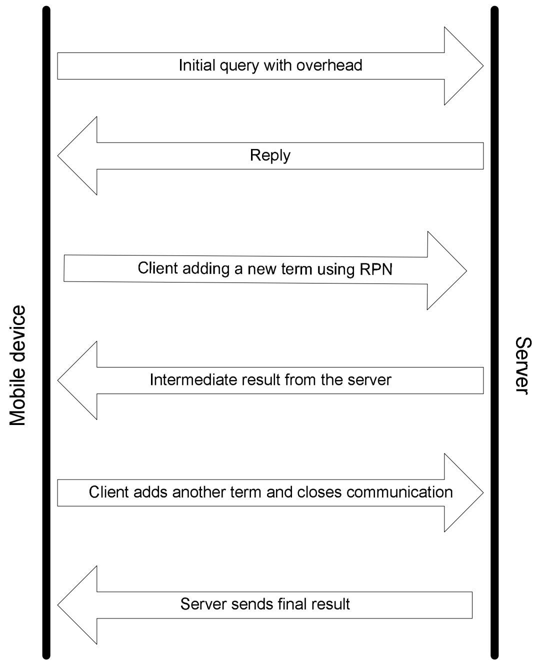
Mobile device
Server
Initial query with overhead
Reply
Client adding a new term using RPN
Intermediate result from the server
Client adds another term and closes communication
Server sends final result

Figure 3. Example of a communication sequence between a mobile device and a server using query streaming. There is only one updateable result set residing in the server at all times.

2.2 USE OF REVERSE POLISH NOTATION

Reverse Polish Notation (RPN) [10], also known as the postfix notation, is a method of arranging an equation whereby the operators follow the operands. This contrasts with infix notation, where the operators are between the operands. The advantage of RPN is that the order of operations is implicit in the equation; hence special operators denoting that order are not required (as with infix notation). Another advantage is that the equation can be processed in-order using a stack.

By using a stack, a multimedia query using RPN can effectively be streamed to a server. This results in a query format for mobile devices that has minimal overhead. At one level this allows simple refinement of queries for query-by-example. However at another level the main advantage of this technique is the ability to perform a search within the previous set of results, as shown in Fig. 3. In Fig. 3, there is one result set at the server and this is updateable by the client. This requires that the client and server have an open communication port for the duration of the query so that the server can send intermediate results to the mobile client. By virtue of RPN, the intermediate results will be correct, even if the client adds complex multiple new terms, since the server does not need to receive the whole query to begin processing. We call this method "query streaming".

Query streaming can be thought of as performing a search using e.g. the iTunes search box, where adding another letter to the search term further refines the query without performing a new query. This is unlike a Google search; where adding or removing a term in a search requires a new search to be performed. For query streaming, however, instead of querying a local database (as in the case of iTunes), the refinement process is performed over the network.

2.3 MQF – A GENERAL PURPOSE MULTIMEDIA QUERY FORMAT

Our previous work in the area of multimedia query resulted in a general purpose multimedia query format called MQF [9]. MQF was designed as a simple container format for use in multimedia query influenced by SOAP [11]. Key strengths of MQF includes the use of RPN notation, the concept of query levels to remove ambiguity regarding the meaning of a query term sent by a client, free-form operators and a simple design to provide extensibility for the future.

In MQF, a query term can take one out of four possible forms:

1. Exact: the term is to be matched exactly by the server (e.g. an MPEG-7 description)
2. Example: the term is an example, where the server is to match it as close as possible to its database.
3. Semantic: the term is a semantic description of a multimedia data. Semantic information means that the information is higher level information and cannot be extracted from the actual multimedia data itself (e.g. creator information, title, etc).
4. Free text: the query term is a natural language based query with no predetermined structure.

MQF uses SOAP-like containers to act as terms in the query and these are then stacked in an RPN sequence to be processed by the server.

2.4 MODIFICATION TO MQF TO ENABLE QUERY STREAMING

To achieve "query streaming" from mobile devices, a new query format with built-in RPN capabilities is required. Existing solutions based on the SQL [12] such as SQL/MM [13] cannot be used for this purpose due to the fact that SQL was designed for textual databases with a rigid, known structure, and each SQL statement must be a complete query. Hence, using SQL, a new query would need to be formulated and cannot be streamed to the server. In contrast, to achieve effective query streaming, the query format must handle data with an unknown structure. MQF [9] is a good candidate for performing query streaming due to its built-in support of RPN and its open format. However MQF was not originally designed for query streaming, therefore some modifications are required to enable query streaming using MQF.

A required modification to MQF is the inclusion of a query streaming flag in the first query. All the proceeding messages from the client (following the first query) must

```
<mqf>
  <queryId>id_123</queryId>
  <from>client.uri</from>
  <query stream="true">
    <replyType>image/jpeg</replyType>
    <item queryLevel="example">
      <mpeg7:MediaData64>
        AaBbCc/==
      </mpeg7:MediaData64>
    </item>
  </query>
</mqf>
```

Modification to enable query streaming for the initial query.

Figure 4. Initial communication from the client.

```
<mqf>
  <queryId>id_123</queryId>
  <from>server.uri</from>
  <replies>
    <reply>
      <item>
        http://server/image1.jpg
      </item>
      <index>1</index>
    </reply>
    <reply>
      <item>
        http://server/image2.jpg
      </item>
      <index>2</index>
    </reply>
  </replies>
</mqf>
```

Figure 5. Server reply.

Modification to enable subsequent queries.

```
<query streamId="id_123">
  <item queryLevel="exact">
    <dc:Creator>John Doe</dc:Creator>
  </item>
</query>
```

Figure 6. Client refinement: adding a term to the query.

also include information signifying that they are part of a query stream. An example is shown in Figures 4, 5 and 6. In Fig. 4, the client initially sends a normal MQF query with the added attribute of *stream=true* in the <*query*> element, signaling the server to expect an XML fragment for the next packet of communication. In Fig. 5, the server then replies as per MQF specification [9]. In Fig. 6, the client then sends a refinement of the query. In this case, the client would like an image created by John Doe based on the server replies in Fig. 5. Note that the client sends only the <*query*> element, with the query identification number as an attribute to let the server know that this packet is part of the specific numbered query stream. This identification number enables a device to potentially open multiple query streams to the server.

By using RPN, the communication sequence shown in Figures 4, 5 and 6 is possible and efficient since the server can process the terms as they arrive on the server in sequence.

3. QUERY STREAMING USING FRAGMENT REQUESTS

3. 1 FRAGMENT REQUEST AND UPDATE UNITS

Fragment Request Units (FRUs) [14], which is part of the MPEG-B standard [15], are created by the users to request fragments of XML from a remote XML document.

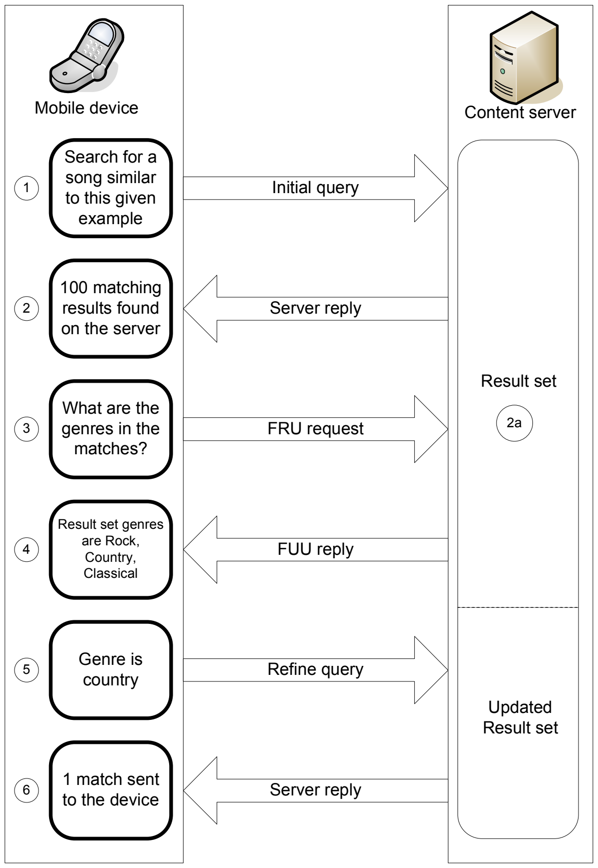
Mobile device
Content server
1
Search for a song similar to this given example
Initial query
2
100 matching results found on the server
Server reply
3
What are the genres in the matches?
FRU request
4
Result set genres are Rock, Country, Classical
FUU reply
5
Genre is country
Refine query
6
1 match sent to the device
Server reply
Result set
2a
Updated Result set

Figure 7. Query refinement using FRU.

```
<mqf>
  <queryId>id_123</queryId>
  <from>client.uri</from>
  <query stream=”true”>
    <replyType>audio/mpeg</replyType>
    <item queryLevel=”example”>
      <mpeg7:MediaData64>
        AaBbCc/==
      </mpeg7:MediaData64>
    </item>
  </query>
</mqf>
```

Figure 8. Initial query: Search for a song similar to this given example (Fig. 7–step 1).

```
<mqf>
    <queryId>id_123</queryId>
    <from>server.uri</from>
    <replies>
        <text>100 results matched</text>
    </replies>
</mqf>
```

Figure 9. Server reply: 100 matching results (Fig. 7–step 2).

```
<mqf>
  <queryId>id_123</queryId>
  <from>server.uri</from>
  <replies>
    <reply>
      <item>
        <mpeg7:MediaUri>
           http://server/1.mp3
        </mpeg7:MediaUri>
      </item>
      <description>
        <mpeg7:Genre>Rock</mpeg7:Genre>
      </description>
      <index>1</index>
    </reply>
          ...
    <reply>
      <item>
        <mpeg7:MediaUri>
          http://server/100.mp3
        </mpeg7:MediaUri>
      </item>
      <description>
        <mpeg7:Genre>Country</mpeg7:Genre>
      </description>
      <index>100</index>
    </reply>
  </replies>
</mqf>
```

Figure 10. Result set residing in the server (Fig. 7-step 2a).

```
<FRU>
  <Query>//mpeg7:Genre</Query >
</FRU>
```

Figure 11. FRU request asking the server to specify the genres in the result set (Fig. 7-step 3).

```
<FragmentUpdateUnit>
  <FUCommand>addNode</FUCommand>
  <FUContext>
    /mqf/replies/reply/description
  </FUContext>
  <FUPayload>
    <mpeg7:Genre>Rock</mpeg7:Genre>
    <mpeg7:Genre>Classical</mpeg7:Genre>
    <mpeg7:Genre>Country</mpeg7:Genre>
  </FUPayload>
</FragmentUpdateUnit>
```

Figure 12. FUU reply showing the genres in the result set (Fig. 7-step 4).

```
<query streamId=”id_123”>
  <item queryLevel=”exact”>
    <mpeg7:Genre>Country</mpeg7:Genre>
  </item>
</query>
```

Figure 13. Client sends query refinement, specifying the server to consider only music from the Country genre and updates the result set (Fig. 7-step 5).

The FRUs are created in XML (valid to the FRU schema) from a selection of commands (known as RXEP [16] commands). Briefly, basic FRU commands are as follows: Src, Query, XMLPull and Stream. These commands allow a client to select a document, query a document, issue XML Pull commands on a document and stream sub-branches of a document (for further details see [14]).

FRUs are capable of requesting any fragment of the XML document (based on fragment size and location),

```
<mqf>
  <queryId>id_123</queryId>
  <from>server.uri</from>
  <replies>
    <reply>
      <item>
        <mpeg7:MediaData64>
          DdEeFfGg1234//==
        </mpeg7:MediaData64>
      </item>
      <index>1</index>
    </reply>
  </replies>
</mqf>
```

Figure 14. The server sends the one matching result to the client (Fig. 7-step 6).

thus providing clients with random access. This allows a client to jump into any node in an XML document, or to simply, “navigate backwards” (such an operation may be entirely client side if previous XML fragments have been cached locally). An example of FRU instantiation is shown in Fig. 11. FRU provides the request mechanism and the responses to FRUs are provided as Fragment Update Units (FUU). These FUUs are specified in MPEG-7 Part 1 [17].

MPEG-7 Part 1 specifies a textual delivery method for delivering multimedia descriptors that are described in XML. This method is known as Textual Encoding format for MPEG-7 (TeM). TeM relies on the principle that an XML document can be divided into smaller XML fragments, which can be reassembled at the client. The local version of the XML document on the client side is manipulated through TeM specific commands [17] sent by the server. TeM is capable of fragmenting an XML document and delivering these XML fragments to a client to recreate the original XML document structure. The fragments are encapsulated in XML using Fragment Update Unit (FUU) elements.

FRU and FUU can therefore work together, with the client sending a FRU request using XPath [18] and receiving the requested XML fragment via a FUU. The client is then able to reconstruct a partial XML instantiation of interest using only relevant information provided by the FUU, instead of receiving the whole XML document from the server (which would be prohibitive for mobile devices).

3.2 COMBINING QUERY STREAMING, FRAGMENT REQUEST UNIT AND FRAGMENT UPDATE UNIT

There is a problem inherent in the scenario 1 described in Section 2.1. What if the user received 100 matches and does not know the genre associated with the song? The user then could not formulate the correct query request update without guessing. Since the mobile user cannot view all 100 results due to the many limitations of mobile devices, querying the result set itself (server side) to formulate a query update became an important feature requirement. By combining query streaming and FRUs, a search could be easily refined on the server using FRUs without actually changing the intermediate result set generated by the query.

Thus a different version of scenario 1 in Section 2.1 could then be: Instead of sending another refinement to the query, the client specifies, using FRUs, that the song in question is part of the “Country” genre. The user can then explore the result set in the server by sending FRU request specifying that only music in the country genre be considered, and instructing the server to check how many songs matches that specification. Consequently, the user performs a secondary search in the server’s result set before performing a query refinement. This FRU capability (to browse an XML tree without knowing the underlying schema) is useful in the case of queries to a server with an unknown metadata description scheme.

An example of the updated use case scenario is shown in Fig. 7, with the associated example XML instantiation shown in Fig. 8-14. The client is a mobile device with limited hardware capability and high bandwidth cost, and the user has no idea of the genre associated with the example given (Fig. 8), and the server specified that it has 100 matches (Fig. 9). The client subsequently sends an FRU request to the server asking it to specify what genres are available in the result set (Fig. 11) and the server replies accordingly using FUU (Fig. 12). The client is certain that what he/she wanted is from the country genre, and thus sends the query update (Fig. 13). The server is left with only one match after the update, and sends the data to the client directly (Fig. 14). The client, as a result, does not waste bandwidth sifting through all the matches returned by the server (which is an inherent problem with query-by-example applications).

4. COMPRESSING QUERY COMMUNICATIONS

While FRUs and FUUs provide an efficient XML solution for query streaming, a weakness of XML is its verbosity size. The size of XML is obviously a problem when dealing with mobile devices with expensive bandwidth, especially considering that any binary data embedded within an XML document must be converted into its ASCII equivalent, thus expanding its size by 33%. MPEG recognized this limitation and MPEG-7 overcomes this limitation by providing a tool to convert XML to a smaller binary format during transport over a network(this is called BiM [17]).

Table 1. MQF compression result using BiM.

Example file	Original XML size (Bytes)			BiM compressed (Bytes)		
	Total size	Binary attachment size (text format)	MQF size	Total size	Binary attachment size (binary format)	MQF size
Figure 8 – initial query with embedded binary	5181	3828	1353	2835	2794	41
Figure 9 – initial server reply	375	-	375	40	-	40
Figure 13 – query refinement	378	-	378	136	-	136
Figure 14 – final server reply with embedded binary	5690	3828	1862	3051	2794	257

BiM is a tree-based compression technique that utilizes knowledge that schema information should be common to all users and, hence, can be regarded as a priori information [19]. The consequence is that all XML files conformant to that schema can be significantly compressed through binary encoding of the tags [20]. BiM operates on the principle that every child node of the schema tree can be assigned a binary code [17]. This allows the binary rather then the textual transmission of the XML tag. BiM also applies datatype specific compression based on the information provided by the XML Schema. If the XML document contains embedded binary data encoded in ASCII, the data is converted to its original binary representation by BiM.

For transport purposes, we compressed the examples in Fig. 8, 9, 13 and 14 using BiM. The results are shown in Table 1. The embedded binary data size in the experiment is 3828 bytes in text format and 2794 bytes in binary format.

The results in Table 1 show that on average, the size of the MQF is 992 bytes. These MQF sizes are compressed by BiM to an average size 118 bytes. Hence, for the four example messages, BiM offers a 46% reduction in data transmission and 86% saving for MQF. However, for longer query and responses, significantly higher savings can be made.

5. CONCLUSION

This paper proposed a new method for performing multimedia query-by-example using the concept of query streaming in combination with FRUs for use in mobile devices with limited bandwidth resources. It is shown that using MQF compressed with BiM, the overhead could be compressed by more than 50%, thus saving expensive mobile bandwidth. MQF, combined with FRUs and FUUs, also provides a flexible multimedia query by content method for mobile devices with limited hardware resources via query streaming and result set examination.

6. ACKNOWLEDGMENTS

The authors thank Gerrard Drury for useful conversations that led to the work presented in this paper.

12. REFERENCES

[1] "YouTube: broadcast yourself," http://www.youtube.com.

[2] "VideoJug - Life Explained. On Film," http://www.videojug.com.

[3] "Flickr: photo sharing," http://www.flickr.com.

[4] ISO/IEC, "MPEG-7 Part 5: Multimedia Description Schemes (MDS)," ISO/IEC 15938-5 2001.

[5] "Dublin Core Metadata Initiative," http://dublincore.org.

[6] IBM, "IBM Multimedia Analysis and Retrieval System (MARVEL)," http://mp7.watson.ibm.com/marvel.

[7] W. Muller, H. Muller, S. Marchand-Maillet, T. Pun, D. Squire, Z. Pecenovic, C. Giess, and A. P. d. Vries, "MRML: A Communication Protocol for Content-Based Image Retrieval," *Visual Information and Information Systems*, pp. 300-311, 2000.

[8] J. Li, M. Ozsu, and D. Szafron, "Moql: A multimedia object query language," *Technical Report TR-97-01, Department of Computing Science, University of Alberta, January 1997.*, 1997.

[9] K. Adistambha, C. H. Ritz, and I. S. Burnett, "MQF: An XML Based Multimedia Query Format," presented at International Conference on Multimedia and Expo, Beijing, China, 2007.

[10] M. McIlroy, "Reverse Polish Notation," From MathWorld--A Wolfram Web Resource, created by Eric W. Weisstein, http://mathworld.wolfram.com/ReversePolishNotation.html.

[11] W3C, "Simple Object Access Protocol," http://www.w3.org/TR/soap, 2000.

[12] ISO/IEC, "Database Language: SQL," ISO/IEC 9075, 1999.

[13] J. Melton and A. Eisenberg, "SQL Multimedia Application Packages," *ACM SIGMOD Record*, pp. 97-102, 2001.

[14] S. J. Davis and I. S. Burnett, "Collaborative editing using an XML protocol," presented at IEEE TENCON, Melbourne, Australia, 2005.

[15] ISO/IEC, "Fragment Request Unit," ISO/IEC 23001-2, 2006.

[16] S. J. Davis and I. S. Burnett, "On-demand partial schema delivery for multimedia metadata," presented at International Conference on Multimedia and Expo, Toronto, Canada, 2006.

[17] ISO/IEC, "MPEG-7 Part 1: Systems," ISO/IEC 15938-1 2002.

[18] W3C, "XML Path Language (XPath) 2.0," http://www.w3.org/TR/xpath20, 2007.

[19] U. Niedermeier, J. Heuer, A. Hutter, W. Stechele, and A. Kaup, "An MPEG-7 Tool for Compression and Streaming of XML Data," presented at International Conference on Multimedia and Expo (ICME), 2002.

[20] M. Cokus and D. Winkowski, "XML Sizing and Compression Study for Military Wireless Data," presented at XML Conference and Exposition, Baltimore, USA, 2002.