

nized interaction both locally and remotely. For example, the iTV Mobile service allows users to perform actions such as voting and personalization (e.g.: related menus or sub-menus, advertising and content relating to an end-user profile or service subscription).

**[0044]** The iTV use case can be described in four steps which correspond to four services and sub-services available in an iTV mobile service. The first service is an iTV profile/menu service having the following sub-services: (1) Mosaic menu: TV Channel landscape; (2) Electronic Program Guide and triggering of related iTV service; (3) actual iTV service; and (4) Personalized Menu “sport news.” The second service can comprise a live enterprise data feed, where stock tickers that stream real-time quotes, live intraday charts that show technical indicators, and news monitoring, weather alerts, charts, business updates, etc. are provided. The third service can comprise live chat. The live chat service can be incorporated within, but not limited to a web cam, video channel, or rich media blog service. End users can register, save their surname and exchange messages. Live chat messages appear dynamically in the live chat service along with rich media data provided by the end user. The live chat service can be either private or public in one or more channels at the same time. End users are dynamically alerted when new messages from other users arrive. Dynamic updating of messages within the live chat service can also occur without reloading a complete page. The fourth exemplary service is karaoke, where a music TV channel or video clip catalog is displayed together with the animated lyrics of a song. The animated lyrics can comprise fluid-like animation applied to the text characters of the song’s lyrics in order to make the text characters appear as though they were being sung along with the song (e.g., smooth color transition of fonts, scrolling of text, etc.) The end user is then able to download a song of his/her choice along with the complete animated lyrics by selecting an interactive button.

**[0045]** As discussed above, an RTP payload is used to describe and/or define an XML data fragment. Therefore, an RTP payload format is also defined. A prior art RTP packet format is shown in FIG. 5a, where a common payload header 524 comprises a type field 540, which indicates the type of payload that content sample/payload 526 comprises, an “A” flag 542, a priority (P) flag 544, and a counter (CTR) field 546. The A flag 542 comprises a single bit field, which when set to one indicates that a packet either is, or contains, a random access point. The P field 544 indicates that priority associated with the payload, i.e., some payloads that have a higher priority may be transmitted on a more reliable channel than that used to transmit a payload of lower priority. The CTR field 546 that is incremented by one for each new packet of corresponding priority.

**[0046]** FIG. 5b shows an RTP packet format used in the various embodiments of the present invention. In addition to the RTP header 522, the common payload header 524, and the payload 526, a fragmentation unit (FU) header 550 which follows the common payload header 524 and a fragment header 552 which in turn follows the FU header 550 are also defined. It should be noted that although it is not shown in FIG. 5b, the common payload header 524 is comprised of the same fields as described above in FIG. 5a. In the case of fragmented packets, the type field 540 is set to 6, indicating that the payload 526 is an FU. In contrast, conventional RTP packets generally have 5 types of pay-

loads defined, where a type 1 packet contains one or more sample description(s), a type 2 packet contains a complete SVG scene sample or one of its fragments, a type 3 packet contains a complete SVG update sample or one of its fragments, a type 4 packet contains a list of SVG elements that are currently active, and a type 5 packet contains sample dissimilarity information.

**[0047]** The syntax of the FU header 550 is also shown in FIG. 5b. A 3-bit sample type field 554 indicates the type of the sample contained in the fragmentation unit. The sample type field 554 can take on one of the following five values: 0 indicates a distributed random access point; 1 indicates a sample description; 2 indicates a scene; 3 indicates a scene update; and 4 indicates sample dissimilarity information. It should be noted that the sample type field 554 do not take values 5, 6, or 7. A 2-bit fragmentation type field 556 indicates the semantics of the fragmentation used in forming the fragmentation units. The fragmentation type field 556 can take on one of the four following values: 0 indicates brute force XML fragmentation; 1 indicates syntactic fragmentation; 2 is reserved; and 3 is reserved. The remaining 3-bit reserved field is reserved for possible future extensions.

**[0048]** The syntax of the fragment header 552 depends on the fragmentation type 556 indicated in the FU header 550. For various values of the fragmentation type, the syntax and semantics of the fragment header are described below.

**[0049]** For each fragmentation-type, the syntax of the fragmentation header will satisfy certain requirements. For a lossless case, where no fragments are lost, the syntax enables a receiver to reassemble the content sample from its fragments when all fragments are received. For a lossy case, when one or more fragments of a content sample are lost, the syntax may allow the receiver to conceal the effect of packet loss on the reassembled sample.

**[0050]** In one embodiment of the present invention, a first type of fragmentation, referred to as brute force XML fragmentation, can be utilized. This embodiment of XML fragmentation involves an arbitrary splitting of XML data based on MTU size without taking into consideration the syntactic structure of the XML content. FIG. 2 illustrates an example of brute force XML fragmentation, where XML content 200 is fragmented into fragments 210, 220, 230, and 240 without regard for where the fragment partitions are made. For example, the element “CD” is partitioned between its “country” and “company” sub-elements, creating the fragments 210 and 220.

**[0051]** When brute force XML fragmentation is utilized, as shown in FIG. 2, the XML data is easily fragmented into its respective fragments, 210, 220, 230, and 240. If all of the fragments 210, 220, 230, and 240 are received by the receiver, the XML content is easy to reconstruct. However, if one or more of the fragments 210, 220, 230, and/or 240 is/are lost, it is difficult for the receiver to reassemble the data because the receiver has no knowledge of the nesting structure of the XML content. In addition, error concealment is also difficult to perform because if fragments are lost, brute force XML fragmentation relies mainly on the retransmission of lost fragment packets.

**[0052]** Three possible options exist regarding a fragment header syntax for brute-force fragmentation: (0) Start flag, End flag; (1) Sample ID; (2)

TotalFragmentsPerSample. They are summarized with their associated advantages and disadvantages in Table 1 and