and **420** comprise 2 or 3 fragments of sample 1, with a priority of 3. The last missing RTP transport packet **430** can be determined by the receiver to belong to sample 2 with priority 2. Because the missing RTP transport packets **410** and **420** of sample 1 have a high priority, each can be retransmitted since only one-third of the content of sample 1 was received by the receiver or XML client. Regarding sample 2, three-fourths of the content of the sample 2 was received, and has a lower priority. The receiver may then choose to simply apply error concealment to sample 2.

[0058] In another embodiment of the present invention, a method of XML fragmentation referred to as syntactic XML fragmentation is utilized, which involves the splitting of XML data based on MTU size. In addition, the underlying syntactic structure of the XML content is taken into consideration. FIG. **8** illustrates how an XML document **800** is partitioned according to syntactic XML fragmentation. It should be noted that the XML document **800** contains nested elements, i.e., "path style" within the "svg" element. Each partition that is to comprise a fragment is denoted by a nesting ID. FIG. **8** illustrates 7 partitions denoted by nesting IDs, 1, 2, 3, 3.1, 3.2, 3.3, and 3*, where the "0.1," "0.2," and "0.3" denote the level of nesting from the parent node or element, and the "*" denotes a corresponding end tag of the parent element.

[0059] If packet loss is experienced when utilizing syntactic XML fragmentation in an embodiment of the present invention, it is relatively easy for a receiver to reassemble XML data without errors in XML document object model (DOM) reconstruction because the nesting structure of the XML content is known. In addition, it is easy to perform error concealment if fragment packets are lost. However, a higher level of complexity is encountered when fragmenting XML data using syntactic XML fragmentation. Moreover, in a scenario where it is known that either all fragments are received by the receiver or very few fragments are lost, syntactic XML fragmentation may be viewed as extra overhead, both for fragmentation and reassembly purposes. In such a case, brute force XML fragmentation may be a preferable approach.

[0060] Referring to FIG. **8**, XML documents often contain elements that appear within another element, i.e., "nesting". Nesting can serve the purpose of keeping order in an XML document. Therefore, an element which is nested inside another element, i.e., a parent node or element, needs to end before that parent element. Hence, in order to construct a DOM correctly, certain rules are generally adhered to, such as elements that are opened first must be closed last. Another applicable rule is that nested elements, i.e., elements that occur in the middle of an XML document, need to be closed before those elements that came before them. FIG. **7** illustrates an example of correct and incorrect nesting arrangements. Example (a) shows that nested element "name" has not been closed before the element "number." Example (b) on the other hand, shows that the nested element "name" has been closed prior to the closing of element "number."

[0061] Where the fragmentation of XML content is concerned, there is a correlation between the above-mentioned nesting properties (i.e., syntactic structure) that the XML content exhibits and correct reassembly of its fragments at a receiver. By having prior knowledge of the syntactic structure of the XML content, the receiver is more intelligent in terms of how the fragments can be re-assembled. Because brute force XML fragmentation does not take the syntactic structure of XML content into consideration, it mainly relies on retransmission for correct DOM reconstruction in the event of packet loss. However, if there is a predictable chance that frequent small-scale packet loss occurs, syntactic XML fragmentation provides a more efficient way of DOM reconstruction. In order to store the appropriate syntactic information with the fragments for transmission and reassembly, various embodiments of the present invention utilize a nesting representation with corresponding nesting IDs. One embodiment of such a representation is depicted in FIG. **8**, as described above.

[0062] Certain rules should be observed as well when utilizing nesting IDs. These include: (1) Only nesting IDs belonging to one sample are stored in each packet. In other words, nesting IDs belonging to different samples are not included in the same packet; (2) For each XML fragment, if more than one nesting ID is stored in the fragment, and all child elements are contained within the parent element, only the nesting ID of the outermost element is stored as the inner content is automatically included. For example, again referring to FIG. **8**, if the XML content represented by nesting IDs 3, 3.1, 3.2, and 3.3 are in the same packet, then only the value 3 is stored as a nesting ID with the fragment packet; and (3) For each XML fragment, if more than one nesting ID is stored in the fragment, and not all the child elements are contained within the parent element, then all of the individual nesting IDs are stored. For example, in FIG. **8**, if only the XML content represented by nesting IDs 3, 3.1 are contained in the same fragment packet, both values, 3 and 3.1, are stored as nesting IDs with the fragment packet.

[0063] As is the case with brute force XML fragmentation, several possible options for the fragment header syntax in syntactic XML fragmentation also exist. These options are summarized in Table 2 with their respective advantages and disadvantages, and illustrated in FIGS. **13***a* and **13***b*. It should be noted that all of the fragment header syntaxes for syntactic XML fragmentation meet the lossy requirement discussed above. It should further be noted that additional bits are used to store the nesting IDs as will be described below.

TABLE 2

Options for Fragment-header syntax for syntactic fragmentation

Fragment header syntax	Description	Overhead	Advantages	Disadvantages
Option 0: Nesting ID	Stores only the nesting IDs with each fragment.	String of varying length	Lowest overhead among the various options for syntactic fragmentation. Helps the receiver in error recovery	Possible for ambiguity in the event of multiple packet loss. Refer to FIGS. 9, 10, and 11 for examples.