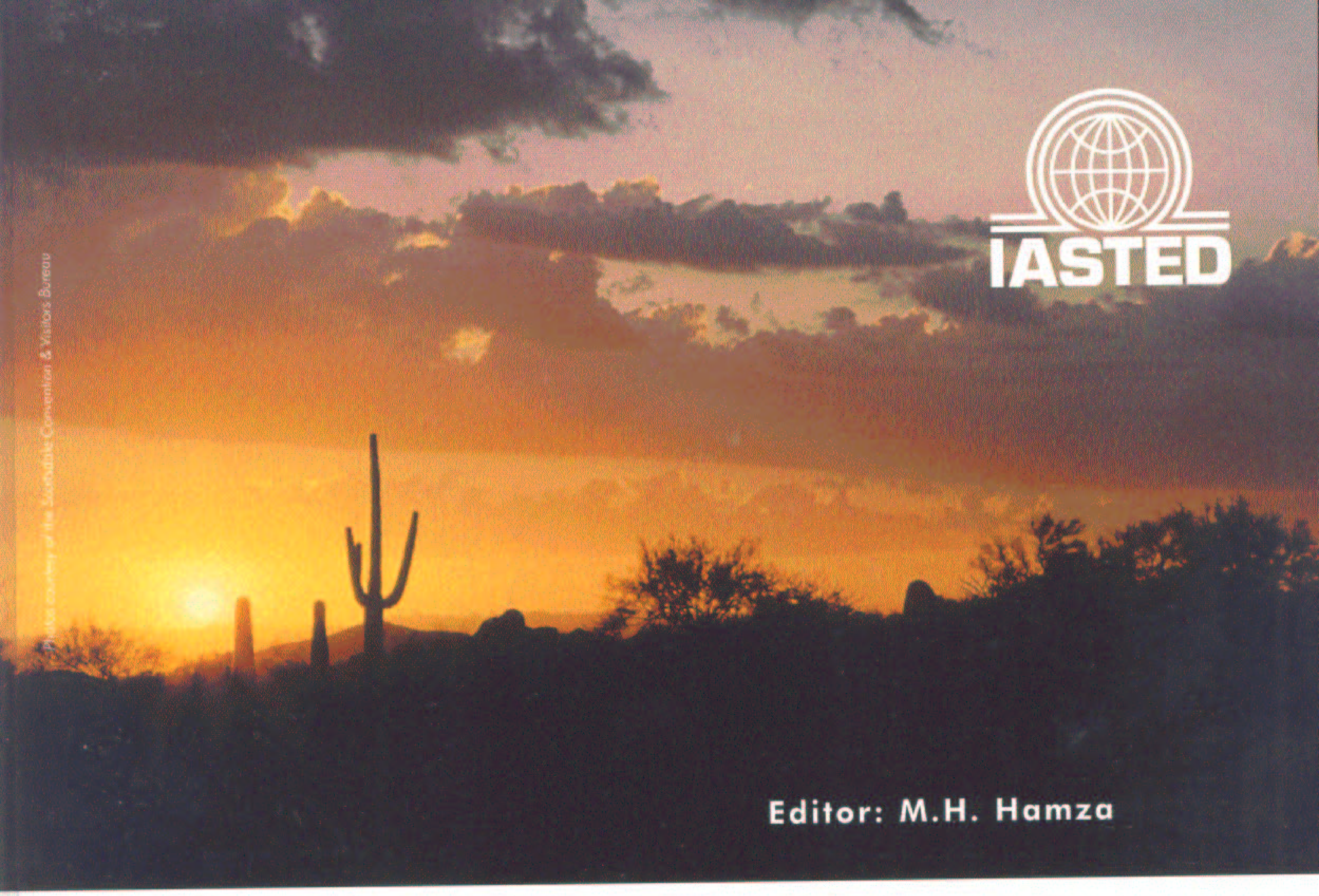

Editor: M.H. Hamza

Proceedings of the Second IASTED International Conference on

COMMUNICATIONS, INTERNET, AND INFORMATION TECHNOLOGY

November 17 – 19, 2003
Scottsdale, AZ, USA

A Publication of The International Association of Science and Technology for Development – IASTED

ISBN: 0-88986-398-9

ACTA Press

Anaheim • Calgary • Zurich

Proceedings of the 2nd IASTEAD International Conference COMMMUNICATIONS, INTERNET, & INFORMATION TECHNOLOGY, Scottsdale, Arizona, November 2003, pp. 553-560,

Transmitting High Quality Archived Object-based Movies with Reduced Bandwidth Requirement

Arvind K. Bansal* and Rahul Pathak
Department of Computer Science
Kent State University, Kent, OH 44242
E-mail: arvind@cs.kent.edu

Abstract

With the exponential increase in the integration of mobile communication devices with media communication over the Internet, the need to transmit high quality archived media clips and movies over limited bandwidth connections has increased. Archived movies have immense use for entertainment such as cartoon movies, news clips, and Internet based mobile instructions. In this paper, we describe an XML based Internet language for frame based transmission of object based movies over the Internet using STMD (Single Transmission Multiple Display) paradigm. The language and its implementation have been described, and the performance results for the implementation have been presented. The implementation benefits from the integration of static analysis of the movie and the predictive buffer management. The benchmark shows that the bandwidth requirement reduces significantly.

Key-words: Bandwidth, Buffer management, Internet, Information, Movies, MPEG-7, Multimedia, QoS, XML

1. Introduction

As the Internet becomes faster and more integrated, the number of archived large multimedia knowledge bases [14] and peer-to-peer communication is increasing. Multimedia clips are retrieved, reprocessed, and transmitted over the Internet in a user transparent way. Recent advances in the Internet transmission standardization using MPEG (Motion Picture Expert Group) [3, 8, 9] and standard Internet based languages such as XML (http://www.w3.org/XML) have made possible ease of movie transmission and cross cultural exchanges of multimedia objects, archived video and audio clips. People are using and demanding the Internet to transmit higher resolution digital multimedia such as multimedia news items, multimedia financial bulletins, music, AV clips, movies, kiosks on mobile platforms such as laptops, PDAs, and devices attached to their means of transportation such as automobiles, airplanes etc. The future society will have a seamless integration of multimedia interaction and multimedia knowledge bases with the Internet.

The advances in the usage and the increasing demand for jitter free high-resolution multimedia clips, has put extra burden on available bandwidth. The rate of increase in the demand for high quality rendering and transmission is much faster than the rate of increase in the transmission bandwidth of the Internet.

In the past, in order to circumvent the problem of bandwidth requirement, many techniques such as archiving the files at the client end cache, compression of video and audio, transmitting frame level delta changes with respect to a reference frame as used in MPEG-4 [3], client end text to speech conversion [4], progressive meshes [6, 10], and XML based modeling of the 3D objects as in VRML97 [16], have been used.

In our previous research [1], we proposed the idea of integrating the notion of hierarchical graphs to model the motion of complex 2D-objects extending the notion of scene-graph based modeling proposed in MPEG-7 [8, 9]. In this model, the image of the subcomponents is superimposed on nodes and edges, and nodes or edges themselves are mapped to an underlying hierarchical graph.

In this paper, we integrate hierarchical graph based modeling, statistical analysis of archived media objects, and predictive sever coordinated client end buffer management. We also extend the model to integrate incremental matrix based background information, and present a grammar and syntax of an intermediate level XML based language called STMDML – A Single Transmission Multiple Display Multimedia Language – for the transmission of synthetic and archived movies.

The major advantages are as follows:

i) The number of transmission of images of the moving components is reduced significantly as it can be archived after the first transmission and retrieved from the client end archive.

ii) Details of a component of the complex object can be stitched to the existing description without a need for the retransmission of the complete object.

iii) Unlike VRML, this approach relies on photo-realistic images thus providing more realism in animation, lighting and shadow effects.

iv) The subsequent frames can be better reconstructed since the data file of the subcomponents is already cached at the client end.

v) Our buffer management system provides a tight integration of server side management of client side cache based upon frame lookahead and static analysis of archived frames. This integration reduces the bandwidth requirement, and reduces the problem of jitter caused by limited processing power of PDAs and facilitates reduced data transfer [2].

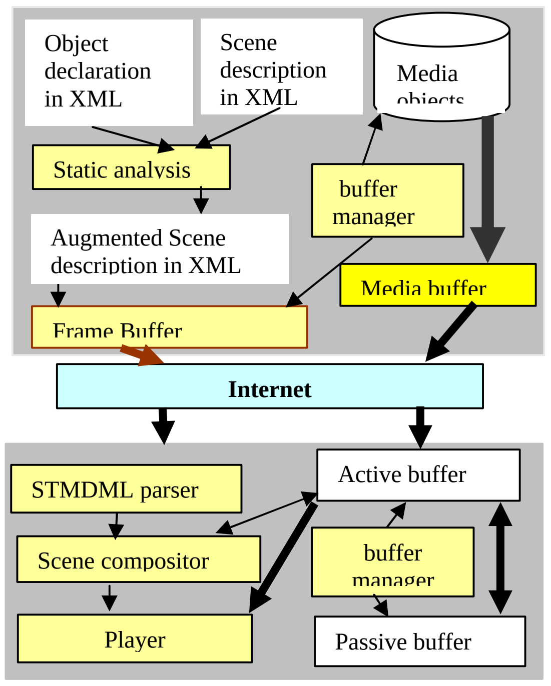

Figure 1: The overall model

The STMDML version 1.0 (including parser, compositor, buffer manager, and renderer) has been implemented using Java and other tool set such as DOM for parsing (http://www.alphaworks.ibm.com /tech/xml4j), Java Swing (http://java.sun.com/products /jfc/) and graphics libraries for rendering.

The paper is organized as follows. Section 2 briefly describes the and relevant features of MPEG-7. Section 3 describes STMDML (Single Transmission Multiple Display Multimedia Language), a representation of complex objects in STMDML, a grammar for the language, and scene representation using the grammar. Section 4 describes an example for XML based representation. Section 5 describes the implementation including a brief description of the parser, the renderer, and the predictive buffer management scheme. Section 6 describes the performance evaluation and compares the work with other popular formats. Section 7 discusses the relevant related works. The last section concludes the work.

2. Background and Definitions

A *hierarchical graph* (possibly directed) [5, 15] has multiple layers of abstraction. Each level of abstraction represents a subgraph (at the lower level of abstraction) having a common attribute/function into a single node. Each node at a higher level of abstraction is a simple node or corresponds to an embedded graph at a lower level of abstraction. The edge between two nodes V_I and V_J represents one or more edges between the embedded subgraphs corresponding to the nodes V_I and V_J. A complex media object can be defined in the form of a hierarchical graph with each *node* of the graph representing a sub-component. An *edge* between two nodes can represent either a relationship between the corresponding sub-components or another component.

MPEG-7 (http://www.chiariglione.org/mpeg/index.htm) [8, 9] has recently become a new industrial standard to represent audio, visual or audiovisual content. MPEG-7 represents a scene as a graph called a scene graph. Every media object or a group of media objects is mapped to a node of the scene graph. Additional information is accompanied with it to provide spatial and temporal synchronization.

3. STMDML Description

STMDML — Single Transmission Multiple Display Multimedia Language ? is a continuously evolving high level XML based language for displaying archived movie over the Internet with an enhanced reuse of components of an object at the client end.

A movie is modeled as a sequence of scenes. A scene consists of multiple frames, and each frame has multiple interacting media objects or group of media objects. Scene representation is a mechanism to spatially locate individual objects and group of media objects that form a scene.

To provide and maintain this realistic environment through out the scene, the background has to be altered at every frame in the scene. During the motion of one or more objects, the background changes incrementally between consecutive frames. Completely changing the background causes jitter due to excessive data transfer, and reduces the QoS. To avoid the drop in the QoS, the background is represented as a matrix of images, and only previously untransmitted matrix blocks are sent.

This reduces the total data that is transmitted and thus improves the **Q**uality **o**f **S**ervice.

3.1 Representing complex objects

Each node in the graph represents a part of a media object. The edges between nodes define relationships between nodes or connecting objects or constraints related to the connected nodes. A *subgraph* or a *complex_object* could be embedded inside a node since each graph is a hierarchical graph. Each *subgraph* has an attribute CG (center of gravity). All nodes within the embedded subgraph use relative coordinates with respect to the center of gravity of the higher level node.

An object is superimposed on a node or an edge. A media object can be of type text, audio, image or complex_object. Multiple media objects are superimposed on a single node (or edge), by embedding multiple object tags within the node tag. This is useful for imposing an image and audio on a single node. A practical example is where a node represents a face of a cartoon object and the audio stream associated with it.

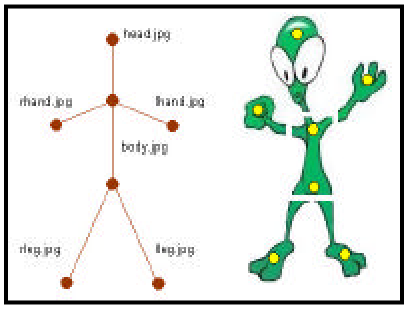

Figure 2. A graph based model of a cartoon

3.2 Scene representation

A movie in STMDML is divided into two parts: '*object description*' and '*scene description*'. The 'object description' is used for storing a library of graph-based templates for objects and association of a symbolic name for the template. The 'scene description' gives frame-by-frame description of a movie. Once a media object is archived at the client end, the subsequent frames refer to the media object by the corresponding identifier.

Frame rate is dynamically altered. The keyword 'total_frames' is used to assign the total number of frames a movie. A *world* groups the entire environment of a movie. The properties associated to a world can be scenes, backgrounds, and media objects. This helps in changing the environment just by referring to a pre-defined *world_id*. The *scene-id* uniquely defines the scene. A loop construct is used to repeat a set of frames. A unique background identifier allows the reuse of the same background in multiple frames or scenes.

The animation of a 2D-object (or group of objects) is modeled by transmitting incrementally (based upon the rendering) only once (and reused later using a client end cache) a finite sequence of images, and background. To facilitate incremental change, background is modeled as a *matrix* of background images. The server dynamically computes the matrix elements needed for the lookahead frame, and transmits the image files for previously untransmitted matrix blocks. Similarly, client associates the matrix element with the correspondingly image file in the cache to retrieve the images.

STMDML uses implicit frame based synchronization to represent temporal relationships. The frames are built statically at the server end before transmitting the movie. A simplified grammar for STMDML has been given in Figure 3.

4. Implementation

The scheme proposed in this thesis is implemented in Java. A parser was implemented using *D*ocument *O*bject *M*odel (DOM), A renderer was implemented using Java and various libraries like swing and graphics and multithreading libraries with Java API. The buffer management was implemented using Java.

4.1 STMDML parser

The STMDML parser is implemented in Java using DOM. Upon receiving an XML frame, the objects are parsed, and a DOM tree is formed. Once a scene node is found it parses the DOM tree further to store all the attributes associated with the node scene using global variables. These global variables are used in scene composition and rendering.

The objects are stored in the object database. The object database consists of tables for objects and frames. The object table stores a tuple for every object. The object tuple consists of {*object_id, object_coordinates, object_source, object_size, object_description*}.

```
<movie> :: '<movie' {<movie_attributes>}* '>'
                            {<scene>}*  '</movie>'
<movie_attributes> :: frame_rate = <integer>
                           total_frames = <integer>
<scene> :: '<scene' world_id = <integer>
                        (scene_id = <integer> '>'
               {<frame>}* '</scene>'
<frame> :: '<frame'  frame_id = <integer>'>'
              {<object>}* '</frame>'
<loop> :: '<loop  [duration  = <integer>] '>' {frame}*'</loop>' |
             '<loop>' [duration = <integer>]{Scene}* '</loop>'
```

```
<object> :: <complex_object> | <background> |<image>|
              <audio> | <text> | <simple_object>
<complex_object> :: '<complex_object' object_id = <integer>
                                  motion = <animate> '>'
                              {graph}* '</complex_object>'
<animate> ::  'moving' |'still'
<graph> :: '<graph graph_id  = <integer>
                       node_count = <integer> '>
              {<graph_elements>}* '</graph>'
 <graph_elements> :: <edge> | <node>
<edge> :: '<edge' edge_id = <integer>
             node_pair = (<integer> <integer>)
             {<edge_attributes>}* '>'
              {<embedded>}* '</edge>'
<edge_attributes> ::  descr = <string> |
                            coord = (<integer>  <integer>)
                                       (<integer> <integer>)
<node> :: '<node' node_id = <integer>
              {<node_attributes>}* '>' {<embeded>}* '</node>'
<node_attributes> ::  descr = <string>
                            coord = <integer> <integer>
                            adj_nodes = ({<integer>}*)
                            edges = ({<integer>}*)
<embedded> :: {<object>}* | {<subgraph >}*
<background> :: '<background'  background_id  =<integer> '>
                    {<background_elem>}* '</background>'
<background_elem> :: '<block'   dimension = <integer>
                                <integer> size = (<integer><integer>)
                                format = <format_type>
                                path = <string>)  '/>'
<image> :: '<image'  image_id = <integer>
                 format = <image_format>
                 coord  = (<integer>,  <integer>)
                 path  = <string>'>'
<audio> :: '<audio'  audio_id = <integer> format =
              <audio_format> [duration = <integer>]
               path = <string> '>'
<text> :: '<text' (text_id  <integer>) (coord = (<integer>
            <integer>) {text_attribute}* '>' <string> '</text>'
<text attributes> :: size = <integer> | color = <color_type>
                           | font  <string>
<format_type> :: <audio_format> | <video_format>
<video_format> ::  'jpeg' | 'mpeg' | 'gif' | 'png'
<audio_format> :: 'wav' | 'avi'
<simple_object> :: '<simple_object'  object_id =<integer>
                           style = <solid> | <circle> | <rectangle>
                           color = <color_type>
                           size = (<integer> <integer>
                          coord = <integer>,  <integer> '>'
```

Figure 3. A simplified grammar for STMDML

4.2 STMDML renderer

The renderer runs at the client side. The frame table consists of data associated to each frame. A frame tuple consists of {frame_id, object1_id, object2_id, object3_id…}. Before the client starts rendering, the frame table is populated with data for the first lookahead number of frames. After rendering the first frame of the movie the frame table is updated by discarding the record for the first frame and adding a record for the new lookahead frame. This continues until the last frame of the movie is rendered.

The display subroutine has code segments to handle every type of object. For example the code to display a complex_object is different from the code to display background. The display subroutine has a switch for render each type of media object. When the display subroutine is called it reads data for the current frame id from the frame table, identifies the object_id for the objects present in the frame, reads data for that object from the object table.

A thread implemented in the renderer calls the display subroutine determined by 1/frame_rate. For example, if the frame_rate = 10 the thread calls the display subroutine every 0.1 second. The frame id is incremented in the thread.

4.3 Predictive look-ahead buffer management

The predictive look-ahead scheme is based upon the concept of static analysis to identify and retain the objects those are present in the look-ahead frames.

In addition to transmitting media files, the server transmits two types of XML frames: *media description frames* and *command frames*. Media description frames are used for rendering the objects, and command frames are used to command client to delete objects, insert objects, or transfer the objects between client-side active and passive buffers. Objects those are not used in near future based upon static analysis and lookahead analysis of frames at the server side are deleted. Similarly, objects those are needed in near future based upon server side static analysis and runtime lookahead analysis are retained in the client side buffers. Objects which will come in future but not in the near future as decided by lookahead frames are transferred from client side active buffer to client side passive buffer.

The initial server side object map is built after the analysis of first *L* frames comprising the initial look-ahead window. After analyzing the first *L* frames, the look-ahead window is shifted by one frame at a time after transmitting the current frame (see Figure 4).

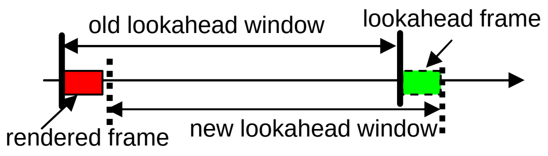

Figure 4. Lookahead analysis for buffer management

After shifting the look-ahead window the counters in the server side object maps are updated. Based upon the commands transmitted by the server, the client deletes, inserts, or transferred objects between passive and active buffer.

The client also uses a backchannel to inform the server if a media object is lost (or delayed) so that server can retransmit the object for composition and rendering in time. This analysis is done after the XML frame is received and analyzed by the client.

Interested readers may find the detailed algorithm for buffer management in [2].

5. An Example

In this section, we describe an example of XML based representation for a simple cartoon movie. In this movie each frame embeds a complex object in it. The images superimposed on the edges and nodes of the graph are changed based on the animation of the complex object.

The code for the scene description for the movie in sync_tazo.xml begins with the <movie> tag definition. The <movie> tag defines the 'frame_rate = 10' and 'total_frames = 40'. This determines the duration of the movie as 4 seconds. The <scene> tag defines the beginning of the scene. The attributes 'world_id = 1' and 'scene_id = 1' define the environment and the scene identifier. The <frame> tag defines the contents of the frame of the movie. The <complex_object> tag defines the complex object.

The attribute 'animation = moving' defines that the complex object is moving. The graph is defined within the <graph> tag. This graph structure defines a complex object. The attribute 'graph_id = 1' defines the graph identifier, and the 'node_count = 7' defines that there are 7 nodes in the graph. The value 'coord = (180, 10)' define the coordinates of the root node. The tags 'edge_id' and 'node_pair' define the node-identifiers and nodes connected by this edge. The attribute value 'descr = head' describes the edge, and the attribute value 'coord = (180, 10), (0, 40)' defines the coordinates of the two connected nodes. Nodes are defined within the <node> tag. The attribute value 'node_id = 1' defines the first node of the graph. The attribute value 'descr = head' describes the superimposed image. The attribute value 'coord = (180, 10)' defines the position of display of the image 'head' within the frame. The attribute 'adj_nodes = 2' shows that the adjacent node has 'node_id = 2'. The attribute 'edges = 1' defines that the edge with 'edge_id = 1' is connected to this node. The 'image_id' of the superimposed image is '1'. The location of the image file is defined by 'path = tazos1_001'. The attribute 'node_id = 2' defines the node. The attribute 'descr = connector' defines no object is superimposed on it.

The attribute 'coord = (0, 50)' defines the position of the node. These coordinates are relative to the center of gravity coordinates. The attribute 'adj_nodes = (1, 3, 4, 5)' defines that this node has four adjacent nodes. The attribute 'edges = (1, 2, 3, 4)' defines that four edges are incident on this node. All the other edges are defined in the same fashion. The graph is terminated by the </graph>. In the similar fashion all other frames in the movie are defined.

```
<!DOCTYPE movie SYSTEM "stmdml.dtd">
<movie frame_rate = 10, total_frames = 40>
  <scene world_id = 1 scene_id = 1>
   <frame id = 1>
    <complex_object object_id = 1 type = moving>
      <graph graph_id = 1 node_count = 7 coord = (180, 10) >
          <edge edge_id = 1 node_pair = (1, 2) descry = head
                   coord = (180, 10), (0, 40)>
             <node node_id = 1 descr = head coord = (180, 10)
                         adj_nodes = 2 edges = 1>
                   <image image_id = 1  path = tazos001_1
                               format = jpg/> </ node>
            <node  node_id = 2   descr = connector   coord = (0,
50)
                 adj_nodes = (1, 3, 4, 5) edges = (1, 2, 3, 4) />
        </ edge>
        <edge edge_id = 2 node_pair = (2, 3) descr = hand
                   coord = (0, 40), (-18, 66)>
            <node node_id = 2/>
            <node node_id = 3 descr = hand coord = (-18, 66)
                       adj_nodes = 2 edges = 2>
                 <image image_id = 2 path =  tazos001_2
                         format = jpg/> </ node> </ edge>
        <edge edge_id = 3 node_pair = (2,4) descr = hand
                    coord = (0, 40),(18, 32)>
            <node node_id = 2/>
           <node node_id = 4 descry = hand coord = (52, 35)
                 adj_nodes = 2 edges = 3>
              <image image_id = 3 path = tazos001_4
                          format = jpg/> </ node> </ edge>
      <edge edge_id = 4 node_pair = (2, 5) descr = body
                   coord = (0,40), (52, 32)>
            <image  image_id  =  4  path  =   tazos001_3  format  =
jpg/>
       </ edge>
        <edge edge_id = 5 node_pair = (5,6) description = leg
                   coord = (52, 32), (-20, 148)>
             <node node_id = 5/>
             <node node_id = 6 descr = leg coord = (-20, 148)
                      adj_nodes = 5 edges = 5>
                 <image image_id = 5 path =  tazos001_5
                        format = jpg/> </ node> </ edge>
        <edge edge_id = 6 node_pair = (5, 7) descr = "leg
                    coord = (52, 32),(35, 148)>
            <node node_id = 5/>
            <node node_id = 7 descr = leg coord = (35, 148)
                            adj_nodes = 5 edges = 6>
                  <image image_id = 6 path =  tazos001_6
                           format = jpg/> </ node> </ edge>
      </ graph> </ complex_object>
 </ frame>
….
   <frame_id = 2>
   . . .
   </ frame>
   <frame_id = 3>
   . . .
   </ frame>
   . . .
</ scene>
…..
</ movie>
```

Figure 5. An example of cartoon movie in STMDML

6. Performance Evaluation

Various performance characteristics in this section show that the model significantly reduces the data transmission requirement thus reducing the jitter at the client end. The factors that are important in transmitting multimedia data over a network are: bandwidth at which the data can be transmitted, total data that is transmitted from the server to the client during a multimedia presentation, and the resolution of the transmitted media objects.

We used a simple cartoon movie using a cartoon character 'Tazo' for the performance evaluation. The 'tazo' cartoon has seven nodes and six edges. The movie has 42 frames. Objects are superimposed on 5 nodes and 1 edge. There are a total of 6 objects superimposed on the graph in each frame. The objects are changing in each frame to create animation in the movie. The object database is 206 KB in size. The size of objects ranges from 5 – 9 KB.

Figure 6 shows a graph for data transferred at each frame against lookahead scope. This is a three dimensional graph with frame id on the x-axis, data transfer in Kbytes on the y-axis and lookahead number of frames on the z-axis. The data shows that the data transfer required at every frame reduces as the movie progresses with the increase in the number of lookahead frames. The data transferred saturates after a certain lookahead number of frames. For this movie the saturation point is 20 lookahead frames. The saturation varies for different movies depending on the theme and the occurrence of animated characters.

Figure 7 shows that total data transmitted to the client reduces with the increase in the number of lookahead frames.

Figure 8 shows a comparison of client clean up at each frame for different buffer space. This analysis is important since mobile devices and PDAs have limited processing power and memory. While, these devices cannot afford extra memory, they will get into jitters if buffer cleaning up goes beyond a threshold. It is observed that the overall clean up time is comparatively less when the client buffer size increases. However, beyond a buffer size, the clean up time for the larger buffer occasionally leads to jitters. For example, for the buffer size 150 KB, the buffer cleanup after Frame 31 is significant.

MRU (**<u>M</u>**ost **<u>R</u>**ecently **<u>U</u>**sed) — one of the popular schemes for buffer management ? retains current objects under the assumption that the recent past is a good indicator of unknown future. The limitation of MRU scheme is that 1) it retains those objects too that will never be used in the future, and 2) gets into jitter problem if the scene changes suddenly.

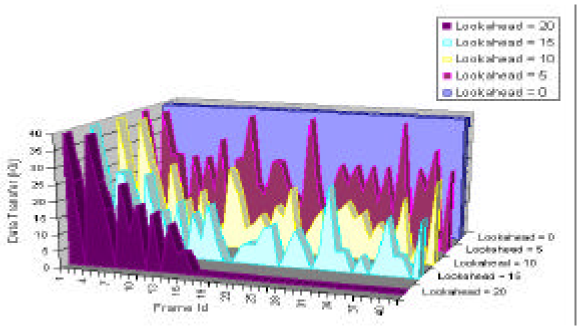

Figure 6. Lookahead analysis vs data transfer

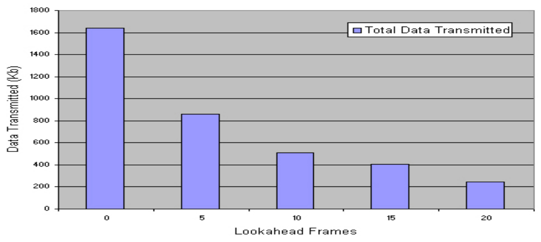

Figure 7. Lookahead vs. overall data transfer in KB

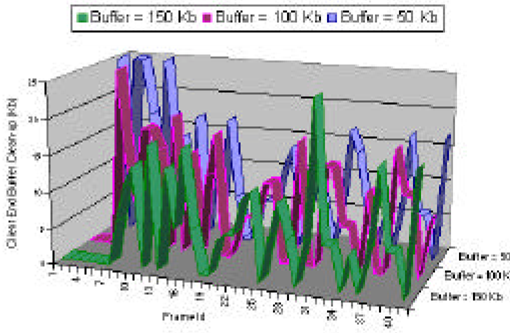

Figure 8. Buffer size vs. buffer clearance rate

We can statically analyze the future objects since we deal with archived movies. The lookahead mechanism retains objects only if the objects are used in the future frames. Figure 9 shows the comparison of the performance of MRU scheme against our predictive push buffer management scheme with a lookahead framesize = 10. The graph clearly shows that data transferred for a lookahead = 10 frames is much less than the data transferred using MRU.

Figure 10 shows a comparison of STMDML with QuickTime, AVI and MPEG formats. The y-axis shows the total data transferred by server. The movie used to generate the data is shown in the appendix. The same movie was converted into quicktime (.mov), avi (.avi) and mpeg (.mpg) formats. A utility networx v2.1 (http://www.softperfect.com/) was used to track the incoming network traffic and to determine total data transferred during the playback of the movie in different formats. All movies were played from the same remote server with very light network load; hence all networking and transmission overheads can be neglected. The bar graph in Figure 10 shows that, despite textual XML based frame representation, the data transferred for STMDML (see the rightmost bar) is much less than AVI (third bar from the left) and MPEG (second bar from the left) and is somewhat better than QuickTime (leftmost bar) for a lookahead frame size = 15.

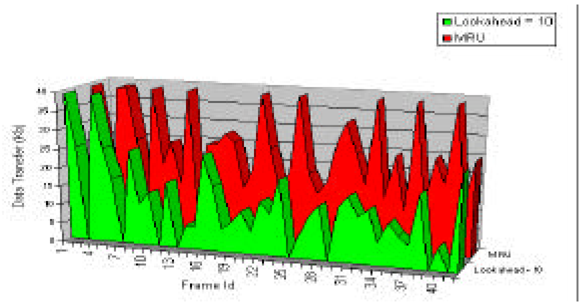

Figure 9. A comparison between MRU and predictive buffer management

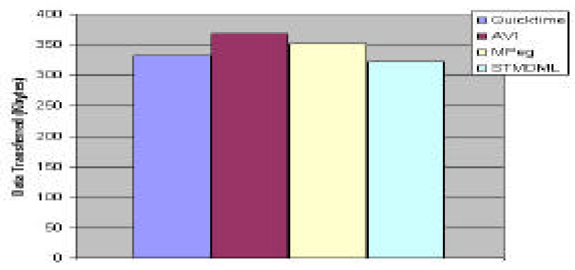

Figure 10. Comparison of data transmission requirement with other popular formats

7. Related Works

Currently, many research works are going on the Internet based delivery of multimedia clips. Most notable are SMIL (http://www.w3.org/AudioVideo/) based extensions, VRML97 based languages such as X3D (http://www.web3d.org/) based extensions such as X3D and Java3D (http://java.sun.com/products/java-media/3D/), MPEG-7 (http://www.chiariglione.org/mpeg/ index.htm) [ ]. Our research benefits from the research on these models, and should be taken as complimentary to these models.

MPEG-7 achieves bandwidth savings as follows: 1) MPEG-7 retains the identity of the moving objects and uses motion vectors to reconstruct the objects in the following frames, 2) MPEG-7 uses MPEG-4's bandwidth saving by predicting frames (P-frames and B-frames) based upon incremental changes (at macro block level) over a reference frame, 3) MPEG-7 uses scene graphs for the reuse of the objects in a scene. However, MPEG-7 does not use hierarchical graphs at the component level. In addition, the loss of reference frames will cause inaccuracies in predicted frames.

VRML97 uses hierarchical graphs and group-objects during object declaration and client end reconstruction to save bandwidth. VRML has many features for virtual reality and 3D modeling. For example, VRML supports event based triggering, collision avoidance etc. However, VRML's approach of using client end object reconstruction using synthetic objects, lacks realism in form, shape, lighting effects, and to achieve near realism, computationally expensive rendering schemes are required. In addition, VRML does not support frame level synchronization. VRML browser also needs additional processing due to 3D modeling which may restrict its use in mobile devices with limited processing power.

SMIL has synchronization constructs to render multiple media objects either concurrently or sequentially using simple programming constructs.

Our model is object based, and has all the benefits of MPEG-7. Our model further enhances MPEG-7 [8, 9] and reduces bandwidth requirement by integrating the objects representation as a hierarchical graph at subcomponent level, by performing static analysis of archived movies for better caching, and by the use of server coordinated predictive buffer management to avoid jitters caused by limited memory and processing power of mobile devices. This enhancement also gives the power to incrementally integrate the detailed image/animation of a smaller focused part of the object with the overall part of the object without retransmitting the image of the overall object.

The scopes of VRML and STMDML are different. VRML is a client end virtual reality language, and is unsuitable for the transmission of multimedia movies due to the lack of frame level finer synchronization. Our aim is to transmit multimedia photo-realistic archived movies (including cartoon movies) over the Internet.

Compared to SMIL based extensions [13], our language is frame based and supports frame level synchronization such as lip synching which is difficult to achieve in SMIL due to low level of tolerance (30 ms) needed for lip syncing.

3D XML based languages such as X3D and 3D tool sets such as Java3D (are still not suitable for mobile devices due to heavy bandwidth requirement for transmitting high quality 3D images even with the availability of innovative concepts as progressive meshes [6]. However, this scenario may change in near future with the introduction of high speed Internet. Even in 3D movies, the implementation techniques and the concepts of STMDML will be very useful in reducing the bandwidth requirement, and these concepts can be built on top of JAVA-3D.

8. Conclusion and Future Work

In this paper, we have described an XML based language for the transmission of 2D cartoon movies over the Internet. The language integrates SPMD (Single Transmission Multiple Display paradigm) [1], static analysis for more accurate retention of objects in the client end cache, and the predictive buffer management scheme based upon tight coordination between a server and the corresponding client to reduce the overall bandwidth requirement. Extensive benchmarks show that data transfer is reduced significantly, and for lookahead frame size = 10, the scheme performs much better than MRU scheme. Similarly, it outperforms other popular format for a lookahead frame size = 15.

Currently, our system can handle only cartoon and synthetic movies due to the unavailability of automated authoring tools. We are developing an automated translator to translate archived movies to object based XML format.

References

[1] A. K. Bansal, T. Kapoor, and R. Pathak, "Extending XML for Graph Based Visualization of Complex Objects and Animation Over the Internet," *Proceedings of the Second International Conference on Internet Computing*, Las Vegas, 2001, pp. 750 - 756

[2] A. K. Bansal and R. Pathak, "Server Coordinated Predictive Buffer Management Scheme to Reduce Bandwidth Requirement for Synthetic Multimedia Movies over the Internet," In *Post-conference proceedings of the Fourth International Conference on Internet Computing*, Las Vegas, to appear, August 2003

[3] S. Battista, F. Casalino, C. Lande, "MPEG-4: A Multimedia Standard for the Third Millennium, Part 1," *IEEE Multimedia*, Vol 6, No. 4, 1999, pp. 74-83

[4] S. Goose, S. Kodlahalli, W. Pechter, R. Hjelsvold, "Streaming Speech: a Framework for Generating and Streaming 3D Text-to-speech and Audio Presentations to Wireless PDAs as Specified using Extensions to SMIL," *Proceedings of the Eleventh International Conference on World Wide Web*, May 2002, pp. 37-44

[5] I. Herman, G. Melançon, M. S. Marshall, "Graph Visualisation and Navigation in Information Visualisation: a Survey," *IEEE Transactions on Visualization and Computer Graphics*, **6(1)**, 2000, pp. 24-43

[6] H. Hoppe, ``Progressive Meshes," *Computer Graphics, In the Proceedings of SIGGRAPH '96*, pp.99-108,

[7] H. Kosch, "MPEG-7 and Multimedia Database Systems," *ACM SIGMOD Record*, June 2002, pp. 644-646

[8] F. Nack and A. T. Lindsay, "Everything You Wanted to Know About MPEG-7: Part 1," IEEE Multimedia, Vol. 6, No. 3, 1999, pp. 65-77

[9] F. Nack, Adam Lindsay: Everything You Wanted to Know About MPEG-7: Part 2. IEEE MultiMedia 6(4), 1999, pp. 64-73

[10] R. Pajarola and J. Rossignac, "Compressed Progressive Meshes," *IEEE Transactions of Computer Graphics and Visualization*, Vol. 6, No. 1, 2000, pp. 79-93

[11] E. Rehm, "Representing Internet Streaming Media Metadata using MPEG-7 Multimedia Description Schemes," *Proceedings of the 2000 ACM Workshops on Multimedia*, November 2000, http://www1.acm.org/sigs/sigmm/MM2000/ep/rehm/index.html

[12] S. Pfeiffer, U. Srinivasan, "TV Anytime as an Application Scenario for MPEG-7," in the *Proceedings of the 2000 ACM Workshops on Multimedia*, November 2000, http://www1.acm.org sigs/simm//MM2000/ep/pfeiffer/index.html

[13] L. Rutledge, "SMIL 2.0: XML for Web Multimedia," *IEEE Internet Computing*, Vol. 5, No.5, 2001, pp. 78-84

[14] S. W. Ryan, A. K. Bansal, T. Kapoor, "A Distributed Multimedia Knowledge Based Environment for Modeling over the Internet," *Proceedings of the International Conference for Tools with Artificial Intelligence*, November 2000, pp. 140-146

[15] I. G. Tollis, "Graph Drawing and Information Visualization," *ACM Computing Survey*, **28 A(4)**, December 1996

[16] K. Walczak, W. Cellary, "X-VRML - XML Based Modeling of Virtual Reality," *IEEE Symposium on Applications and the Internet*, Nara City, Nara, Japan, pp. 204

[17] R. Wolfe, J. L. Lowther, C. K. Shene, "Rendering + Modeling + Animation + Postprocessing = Computer Graphics," *ACM SIGGRAPH Computer Graphics*, Volume 34 Issue 4, November 2000