uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

u Ottawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Firas Kazem

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.A.Sc. (Electrical Engineering)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

An Interactive Streaming System for Image-based Virtual Environments over Heterogeneous Wired-Wireless Networks

TITRE DE LA THÈSE / TITLE OF THESIS

A. Boukerche

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

S. Majumdar

J. Zhao

Gary W. Slater

LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES /
DEAN OF THE FACULTY OF GRADUATE AND POSTDOCORAL STUDIES

An Interactive Streaming System for Image-based Virtual Environments over Heterogeneous Wired-Wireless Networks

By

Firas Kazem

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the MASc degree in Electrical Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering
School of Information Technology and Engineering
University of Ottawa

Your file Votre référence
ISBN: 0-494-14915-9
Our file Notre référence
ISBN: 0-494-14915-9

Abstract

Recent advances in wireless networking and multimedia systems have opened new frontiers in the design of future generations of wireless multimedia systems while creating a new set of trade-offs. The limited bandwidth of wireless networks, along with their varying nature, poses a significant obstacle when interactively streaming the large volume of data that represents image-based virtual environments. In order to overcome this obstacle, we propose an interactive streaming system over wireless networks.

The main contributions of this thesis are:

1. Design and implementation of a system architecture for interactive streaming of image-based virtual environments over wireless networks based on the client server paradigm.
2. Development of an interactive streaming protocol for the purpose of exchanging interactive requests and responses between streaming server(s) and clients.
3. Proposal of a real-time transport protocol (RTP) over wireless networks with a unique payload format and packetization scheme.
4. Proposal of an adaptive rate control algorithm, an immediate feedback mechanism, and a path prediction and pre-fetching scheme to make the streaming process more adaptable and coherent to the changes in network status, and to fully utilize the available bandwidth of the wireless network.

We present our architecture and algorithms, their implementation, and an evaluation of their performance using an extensive set of simulation experiments.

Acknowledgements

I would like to give my thanks to three individuals who helped me fulfill this work:

Dr. Azzedine Boukerche for his supervision and trust in my abilities;

Dr. Tingxue Huang for his guidance, assistance, and constant encouragement;

and Mr. Richard W. Nelem Pazzi for his willingness to discuss my ideas and provide useful suggestions.

This work was partially supported by NSERC, Canada Research Chair Program.

Finally, a special thanks goes to my family; without their continuous sacrifice and support, I would not be where I am today.

Content

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we discuss the status of current technologies pertaining to virtual environments and wireless communications. We examine the prospects of streaming image-based virtual environments over wireless networks by studying the unique characteristics of the medium and by accurately determining system requirements and problems. We finally propose our solution.

1.1 Streaming Interactive Virtual Environment over Wireless Environments

Interactive virtual environments have gained considerable attention in recent years with the emergence of a variety of applications ranging from virtual marketing to network gaming. In the meantime, the rapid development of wireless technologies has led to the emergence of many multimedia wireless terminals such as mobile phones and PDAs that have processing capabilities to deal with high-quality graphics at a satisfactory rendering speed. Combined, these factors highlight the prospects of achieving an interactive virtual environment streamed over wireless networks.

1.1.1 Image-based versus Geometry-based Virtual Environments

In interactive virtual environments, the virtual world is stored on a server(s), with clients retrieving the model for participation and interaction. A virtual environment is

stored as a set of 3D objects. In general, two main approaches to represent a virtual environment exist: an image based approach and a geometry based approach.

The geometry-based approach consists of building 3D models for all the objects within the virtual environment. Applications such as video games can be modeled for 3D representation using modeling languages such as VRML (Virtual Reality Modeling Language). This requires a very complex construction of the virtual world. In addition, as the scene becomes more complex, the task of constructing it becomes more difficult and consumes more computational power. On the other hand, geometry-based methods allow a significant degree of freedom when it comes to interacting with the 3D model.

The image-based approach has the benefit of being passive to the complexity of the scene. This approach also provides a photorealistic representation of the virtual world. Image-based methods allow ease of construction, as this will be done from a set of images taken from various viewpoints. Such representations of image-based 3D scenes have a huge amount of data. The acquirement system is generally a huge camera array like the camera mobile array in Carnegie Mellon University [7] or the Stanford multi-camera array [8].

1.1.2 Virtual Environments over Wireless Networks

When streaming interactive virtual environments over wireless environments, two points must be considered: the fluctuating bandwidth of wireless networks and the limited storage space and computational power of mobile devices. Despite technological advances, wireless terminals lack the resources to handle the complex processing associated with geometry-based virtual scenes. Image-based virtual environments shift the computational load to the server, with the client left only with the task of

decompressing and displaying, as well as recording any interaction done by the user, which is to be sent to the server for processing. Despite the large amount of data represented in a 3D scene, the proper management of the network bandwidth can lead to the efficient streaming of 3D data.

1.2 System Requirements for Interactive Streaming of Image-based Virtual Environments over Wireless Networks

In order to successfully stream interactive image-based virtual environments over wireless networks, a process of four main components must be undergone.

- ***Data Acquisition and Representation:*** The virtual environment is an image-based representation of the real world; its capture and representation is the first step.
- ***Rendering of Data:*** In order to properly display the virtual environment to users, a rendering mechanism must be utilized.
- ***Compression of Data:*** Due to the huge amount of data present in the 3D scene, a compression algorithm must be employed.
- ***Interactive Data Streaming:*** An interactive streaming system is required to transfer the appropriate images to construct the virtual environment.

All four components are critical to achieve the described goal; however, the main focus of this thesis is the design and implementation of the Interactive Streaming System.

In PARADISE Research Laboratories at the University of Ottawa, Project LIVES: Live Interactive Virtual Environment Streaming was initiated in order to reach the described goal. All of the components listed above, including the interactive streaming system, have been implemented as part of this project.

In order to get a better understanding of the interactive streaming system, we briefly describe the first three components as they are all related.

1.2.1 Virtual Environment: Acquisition, Representation, and Reference

Our goal is to provide a set of mobile users with the capability to navigate smoothly through a virtual environment. The virtual environment is represented as a set of 3D scenes stored in an image database. Acquisition of these images can be done by a huge camera array like the camera mobile array in Carnegie Mellon University [7] or the Stanford multi-camera array [8]. A world coordinate system is used to reference these images; for every position and orientation in the system there is a corresponding image.

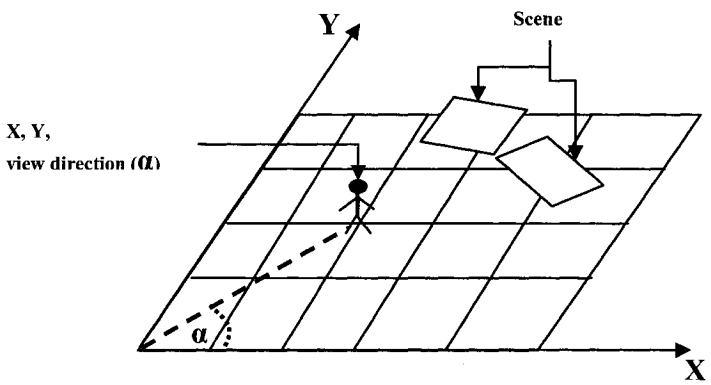

Figure 1.1: World Coordinate System used to reference a set of 3D scenes

Since our system aims to stream over wireless networks, we limit the data acquisition procedure in order to reduce transmission and computational requirements. The position of the person in the coordinate system is represented by two coordinates (x and y), and we assume that the head of the person only turns in the horizontal plane (meaning only to the left or right). Thus, we represent the rotation matrix with a single angle coordinate and reduce the extrinsic camera parameters to three (*x, y, view direction*). We do not limit any intrinsic camera parameters and deform factors.

As mentioned earlier, the virtual environment is represented by a set of 3D scenes stored in an image database containing thousands of 2D images. These original images are taken using different view parameters, including focal length, view-direction, and viewpoint. They provide accurate extrinsic and intrinsic camera parameters for the reconstruction of 3D scenes.

1.2.2 Rendering Through image Morphing

Morphing is an image-based approach for rendering the 3D world for virtual reality systems. Morphing begin by taking two images of the same object from different angles. The next step is establishing the correspondence between the two images. The correspondence between two successive images can be pre-computed and stored as a pair of morph maps. Using these maps, corresponding pixels are interpolated interactively under the user's control to create in-between images [49]. Thus, the second step is simply the process of using mapping to interpolate the shape of each image toward the other. This way, we are able to model a 3D object from a set of 2D images. Morphing as an extraction technique has both its limitations and benefits. In addition to the benefits of being an image-based technique, it allows for inexpensive computations and provides a

smooth quality-speed tradeoff. It also limits the amount of overhead, which is a great advantage for low bandwidth networks. Morphing also gives a real life model of the objects since it is based on actual images of the object. On the other hand, morphing, when compared to geometry-based methods, does not allow a significant degree of freedom when it comes to manipulating the 3D model. In conclusion, morphing is more suitable for virtual environments that are transported over limited resources.

As mentioned earlier, our interactive streaming system adopts image-based 3D scenes. On the server, there is a collection of original images is taken using different view parameters including focal length, view-direction, and viewpoint. On the client, any novel image of a certain view is rendered through morphing two or more original images. This principle is presented in [18], [20], [29].

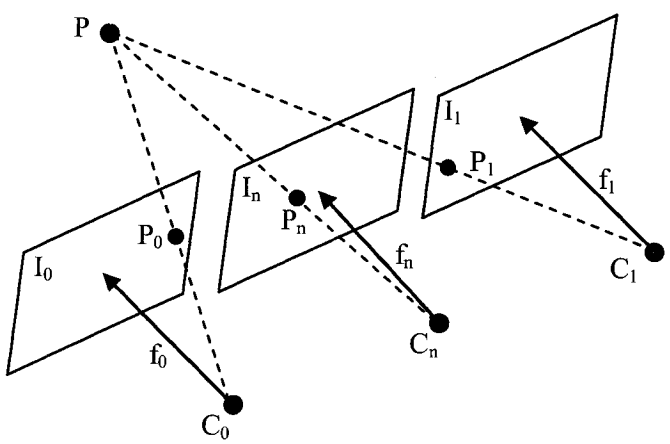

Figure 1.2: Morphing Parallel Views

Any new image of a certain view can be rendered through morphing two or more original images. Since only linear interpolation is performed without pre-warping and post-warping, morphing parallel views proves to be a simple algorithm. As Fig. 1.2

shows, images I_0 and I_1 are acquired at points C_0 and C_1 respectively using corresponding focal lengths f_0 and f_1. The novel image I_n with focal length f_n at point C_n is rendered by interpolating images I_0 and I_1. In short, image morphing provides the capability of rendering a new image of a certain view from two original images. It also provides reusability of original images since adjacent novel views can be morphed from a common set of original images.

The novel image can be rendered on the server or the client. The latter scheme is superior to the former because abutting novel images are possibly rendered from certain common original images. These original images are cached on the client temporarily, in order to be reused. Reusability can reduce transmission traffic and enhance the robustness of the streaming system. In addition, the server is left only with arranging communication sessions and sending images. The server's workload is therefore lessened, and the capacity of the streaming system can be improved.

1.2.3 Compression Algorithm for Image-based 3D scenes

A video has spatial redundancies existing within the same image, and temporal redundancies between two adjacent images. Removing these redundancies is the basic principle of all compression algorithms. There are three different types of encoded frames: I-frame, P-frame, and B-frame. I-frame is an "intra-frame" that removes spatial redundancy through the 2D discrete cosine transform (DCT) algorithm. I-frame is the basic type of frame in video compression. Independent of any other encoded frame, an I-frame contains all information for decoding and displaying. P-frame and B-frame stand for "predictive frames" and "bidirectional frames" respectively. P-frames and B-frames stem from I-frames and P-frames using movement prediction and movement

compensation. P-frames use forward prediction while B-frames use bidirectional prediction. Each type of frame has its own characteristics: I-frames have low compression efficiency but have low disposition delay, P-frames and B-frames have a high compression ratio but cannot be decoded if certain basic I-frames or P-frames are damaged or lost because of their dependence.

Two of the most used compression techniques are MPEG and JPEG. Compared with MPEG, a JPEG video stream includes only I-frames, whereas MPEG makes use of all three types of frames. JPEG has a low compression efficiency and low compression delay. JPEG images can be located very quickly, and the standard is suitable for error-prone environments because I-frames do not depend on any other frame.

As far as a compression algorithm for our system, we make use of Morphing-JPEG. Similar to JPEG, the Morphing-JPEG compressed video consists of pure I-frames for high robustness. However, the compressed object of Morphing-JPEG is a collection of images taken from 3D scenes. These images take the depth information and parameters about viewpoint and view-direction (camera parameters). The novel image is rendered on the client and the requested original images for rendering are transmitted from the server. In general, the requested original images are not neighboring. The server fetches requested original images from different positions in the codestream and then transmits them. The mechanism is different from MPEG and JPEG, which transmit a continuous sequence.

Because recent original images are cached on the client, there are three cases for transmitting original images for the current rendered image: a pair of original images, one original image, and no original image. For example, in order to render the current image,

images 1 and 5 are required to transmit because they are not in the client's memory. For the next rendered image, images 1 and 7 are required but image 1 is already cached on the client. Therefore, image 1 is not transmitted, and only image 7 is sent. If the next rendered image stems from images 5 and 7, no original images are transmitted. This reuse mechanism can save wireless bandwidth resources.

Morphing-JPEG has several unique characteristics. First, the encoded stream is constructed from a continuous series of JPEG still images. Any single encoded image can be located quickly and exactly in terms of acquisition parameters. The requested original images are transmitted in a streaming unit consisting of one or two encoded images. The second characteristic is portability. Acquire all original images is a tedious task. Any conventional compression algorithm demands the acquisition all photographs and then compresses them. Morphing-JPEG can add any image gradually at any time because the image is acquired using a common coordinate system. The third characteristic is to make use of layer-transcoding technology. Each encoded image has one basic layer and several enhanced layers. If the wireless bandwidth is very limited, only the basic layer is transmitted.

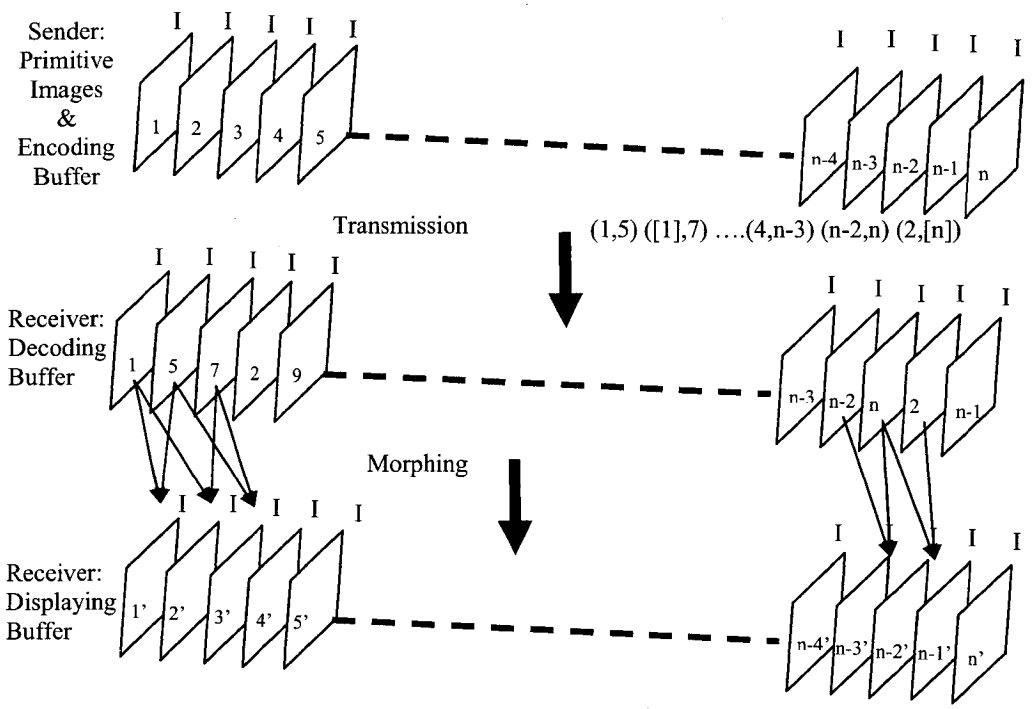

Figure 1.3: Transmission of Morphing-JPEG Compressed Data.

1.2.4 The Interactive Streaming System

Having described the first three components, the remainder of the thesis will focus on our contribution of an interactive streaming system for image-based virtual environments over wireless networks. The following sections will accurately pinpoint the problem at hand, as well as our suggested solutions.

1.3 Problem Statement

When streaming interactive image-based virtual environments over wireless networks, a direct conflict emerges between the huge amount of data present in the 3D scene and the limited bandwidth and fluctuating nature of wireless networks. Interactivity also adds restrictions to streaming, making it an on-demand process based on the client's navigational path in the virtual environment. The server remains unaware of the amount of data required for streaming and the time at which it is requested. This interactive discontinuous streaming process, which runs on the limited resource wireless network, leaves little or no space for retransmissions and buffering.

1.4 Motivation

In order to provide a set of mobile users with the ability to navigate through a virtual environment in real time, an interactive streaming system with client/server architecture is required. A real time streaming protocol with a unique payload format and packetization scheme is needed to account for the image-based view morphing rendering algorithm. In order to cope with the varying nature of wireless networks, an immediate feedback mechanism is required to constantly determine network status. An adaptive rate control algorithm is also essential to adjust the streaming process based on the network status in order to maximize the throughput. Since the data to be streamed depends on the client's navigation in the virtual world, a path prediction and pre-fetching scheme is required to deliver a set of images that the client is predicted to request, and from which the rate control algorithm can selectively stream if the network status deems it feasible. In

order for the streaming process to be interactive, a signaling protocol is needed to establish and control one or more streaming sessions.

1.5 Main Contributions

The main contributions of this thesis are:

1. Design and implementation of a system architecture for interactive streaming of image-based virtual environments over wireless networks based on the client server paradigm.
2. Development of an interactive streaming protocol for the purpose of exchanging interactive requests and responses between streaming server(s) and clients.
3. Proposal of a real-time transport protocol (RTP) over wireless networks with a unique payload format and packetization scheme.
4. Proposal of an adaptive rate control algorithm, an immediate feedback mechanism, and a path prediction and pre-fetching scheme to make the streaming process more adaptable and coherent to the changes in network status, and to fully utilize the available bandwidth of the wireless network.

1.6 Thesis Outline

The remainder of this thesis is organized as follows:

- Chapter 2 presents related work on image-based virtual environments, real time streaming protocols, and rate control mechanisms for wireless networks;
- Chapter 3 presents the design and functionality of our streaming system as well as our path prediction and pre-fetching scheme;

- Chapter 4 presents our design for an interactive streaming protocol (ISP);
- Chapter 5 presents our real time protocol (RTP) with its payload format and packetization scheme, as well as our integrated mechanisms of rate control and immediate feedback;
- Chapter 6 presents a series of experimental results to verify our claims;
- Chapter 7 concludes the thesis and provides some directions for future research.

Chapter 2

Related Work

Our work is associated with three research areas: *image-based virtual environments*, *the real time transport protocol*, and *rate control for streaming over wireless networks*. Although previous research has been conducted in each area, few researchers focus on the combination of all three; mainly the interactive streaming of image-based virtual environments over wireless networks.

2.1 Image-based Virtual Environments

In the early years, geometry-based 3D scenes were the dominant technology [14]. Although geometry-based 3D models can be reused efficiently, the computation complexity of rendering a novel image is proportional to the scene complexity. Recently, the technology of image-based 3D scenes has emerged with the ability to render photo-realistic impressions. Moreover, its computational complexity is independent of scene complexity and is proportional to the image resolution. The only setback is the tremendous amount of data present in a 3D scene. Table 2.1 lists the principal differences between the geometry and image-based approaches.

Table 2.1: Two technologies for 3D Scenes

Parameters	Geometry-based 3D Scene	Image-based 3D Scene
Computing complexity	Proportional to scene complexity	Proportional to image resolution
Data volume	Huge	Very Huge
Rendering speed	Slow	Fast
Reality	Virtual	Photo-Realistic
3D objects	Geometric	Images

To date, a great deal of research has focused on how to represent, compress, dispose, and render image-based 3D scenes. These technologies are based mainly on the famous Plenoptic function [13], [15], [24] through constraining different conditions. For example, light field/lumigraph is a 4D subfunction of the Plenoptic function that assumes that the air is transparent, the radiances along a light ray through empty space remain constant, and the scene is static. Images, panoramas, and concentric mosaics also stem from the Plenoptic function using similar constraints. Such representations of image-based 3D scenes have a huge amount of data and a relatively high computational complexity. Because of these factors, less previous research has been aimed at streaming image-based 3D data over wired-wireless communications in real time.

An immersed system called QuickTime VR [12] is presented. It divides the environment into many grids. A panorama; a wide-angle representation of a view, is taken on each intersection node. When the user "walks through" scenes, the user must "stand" at an intersection node and cannot obtain any rendered image until the whole panorama has been completely downloaded to the client. Our proposed streaming system will outperform the QuickTime VR system on two main points: first, the "walk path" is

not discrete but continuous; second, real-time streaming and rendering replace rendering after transmitting the whole scene data.

[54] designed a mechanism for browsing though large JPEG 2000 compressed images over the Internet. When browsing images, only the portion of the compressed bitstream, according to the client's region of interest (ROI), is downloaded in a progressive fashion to the client side with certain spatial and resolution constraints. This mechanism is designed for one large image and is not suitable for streaming a set of images.

Interactive streaming of high-resolution panoramic views is presented in [6]. It is similar to QuickTime VR except that the data is interactively streamed from the server. [6] accurately defines the key tasks for streaming image-based interactive virtual environments: representation, coding, streaming, and displaying of data. The huge amount of data highlights the need for high compression. However, predictive coding schemes cannot be used because of the user's freedom in navigation. Unlike the timeline and path of movement in video coding, the user's navigational path cannot be encoded into the data. Thus, scene representation is developed containing the image modeling information using MPEG-4 BIFS (Binary Formats for Scenes) based on VRML. Display on the client is done using the HHI 3-D MPEG-4 player. Streaming is based on the MPEG Delivery Multimedia Integration Framework (DMIF). [6] also recognizes the need for a pre-fetching strategy since the delay between the time at which data is requested and the time when it is streamed can hinder fluent navigation through a 3D scene. However, no mechanism to achieve this is proposed. Our system meets the requirements set forth in [6] by presenting a compression algorithm based on JPEG 2000.

It also adopts image morphing for rendering and presents a pre-fetching technique which [6] failed to do this. Finally, it raises the bar by streaming over wireless networks where bandwidth is limited and varying in contrast to streaming over the Internet, as was done by [6].

In [5], a pre-fetching algorithm is proposed. Image samples are positioned on a uniform grid on a plane. The plane is partitioned into square regions, with the image in the center of every block intra-coded and all other images predicatively coded. Assuming continuity of the user's path allows for prediction of which stored images might be accessed next and the organization of data transfer to allow for the most efficient use of the channel bit rate. However, the error prone wireless network is not a suitable medium for applying this pre-fetching mechanism. Because of their dependence, P-frames cannot be decoded if certain basic I-frames or P-frames are damaged or lost. Thus, the proposed mechanism is unsuitable for wireless networks.

In [53] an adaptive panoramic video streaming scheme for image-based rendering was presented. The proposed streaming scheme utilizes panoramas as rendering primitives of their system. Transmission is done through two channels: one for data, and the other for control. JPEG 2000 and JPEG [23, 16] are the compression techniques applied to the raw data. Since the system employs RTP, a rate control mechanism is required. Probing the mobile network's status is followed by proper adjustments to the sending data to avoid network congestion and jitter. Cache management is also considered for efficient memory usage.

Our system bears much resemblance to the one presented in [53]; however, it differs in a number of ways. First, our representation of the real world does not involve

panoramas, which makes the viewing process more realistic and accurate. This point is crucial since the system presented in [53] suffers when motion is translational. A significant variation in frames per second (fps) arises between rotational and translational motion since the latter involves the transmission of a totally new panorama. In theory, our system provides the same performance for both types of motion.

Considering the experimental results of [53], we note that the results were captured for only one mobile client connected to a wireless network with 11 Mbps capacity. Performance shows that the system was able to reach 4~8 fps for JPEG 2000 and 7~14 for JPEG. The variation between the two results is due to the significantly more expensive decoding process of JPEG2000. The wide range of each result is attributed to the variation in performance for rotational and translational motion. Our system not only provides the same performance for both types of motion, but it also outperforms the JPEG 2000 results of [53]. We argue that the results obtained for JPEG in [53] will not hold when dealing with multiple clients. In addition, on a larger scale our system outperforms [53] since it provides a constant 10 fps for up to 15 simultaneously streaming clients.

2.2 Real-Time Streaming Protocol

In traditional video streaming, real-time communication protocols, including real-time transport protocol (RTP) [21] and real-time streaming protocol (RTSP) [23], are utilized. Real-time transport protocol (RTP) is widely applied to all kinds of real-time communications such as simple multicast audio conference, audio/video conference and mixers/translators. It provides end-to-end delivery services for data with real-time characteristics, such as interactive audio and video. Those delivery services include

payload type identification, sequence numbering, time stamping, and delivery monitoring. One important part of RTP is Real-time control protocol (RTCP), which provides periodic monitoring reports of data delivery, minimal control, and identification functionality. RTCP indicates the periodic quality of service.

Certain contemporary real-time transport protocols are for continuous video, but others are for still images. In terms of different communication networks, applications, and services, many video compression algorithms have been developed. The H.261 compression algorithm [28] is adopted to communicate over high-speed networks (>64KB/s), such as DDN and BSDN. The H.263 compression algorithm [29] is applied to low-speed networks (<64KB/s). The MPEG2 compression algorithm [17] is used to supply a high quality video through wide-band networks such as ADSL and CD disc storage. Their compressed objects are videos composed of a series of images whose sequence depends on the position and motion of the photographer. Moreover, two adjacent images have temporal correlativity. For each video compression algorithm, a private RTP [16], [27], [30] was made by the IETF community. Two JPEG compression algorithms [11], [14] were made for a still image. Because different compressed images are completely independent, full intra-frame compression is performed in JPEG algorithms. The RTP for JPEG algorithms was developed [11], [14] for streaming applications of JPEG images. For our proposed streaming system, we presented a specific compression algorithm called Morphing-JPEG through enhancing the JPEG algorithm. Since the real-time transport protocol is unique for every kind of media and each compression algorithm, it is necessary to develop an RTP for the Morphing-JPEG, just as is the case with other compression algorithms. However, this RTP is different from

contemporary RTP protocols. Table 2.2 compares these real-time transport protocols (RTPs).

Table 2.2: Comparison of RTP for H.26x, JPEG, and Morphing-JPEG

Characteristics	RTP for H.26x	RTP for JPEG	RTP for Morphing-JPEG
Correlativity between adjacent images	Continuous or Temporal Correlative	Independent	Independent
Camera Parameters	No	No	Yes
Mechanism for error resilience	No	No	Yes
Monitoring Mechanism	RTP control Protocol (RTCP)	RTP control Protocol (RTCP)	Immediate feedback and RTCP
Image Accessibility	Access images in fixed order	Access any subset of images	Access any subset of images

2.3 Rate Control Mechanisms for Streaming over Wireless Networks

There are generally two approaches for rate control for wired networks: the additive increase and multiplicative decrease (AIMDS) approach and the equation based congestion control approach [4]. TCP friendly rate control (TFRC) is the most popular protocol of the latter approach [4]. Although earlier studies [4] have shown that TFRC performs better than AIMDS, both approaches suffer when applied to wireless networks [2, 4]. Both treat packet loss as an indication of congestion and do not take into consideration the unreliable nature of wireless links. Although loss differentiation

algorithms have been proposed [1], these approaches remain unsuitable for our system, which requires an on-demand interactive streaming of independent images. Rate Control schemes [1, 2] that base their algorithms on an estimated round time trip (RTT) function poorly on wired-cum-wireless networks because of the variation in the RTTs from one client to another based on their distance from the server, and the variation in RTTs from one packet to another depending on the different routes they might take.

Wireless TCP (WTCP) has been proposed, however it targets only the last hop of the network [3]. In [3] a good technique for measuring network status has been proposed relying on the observation that the inter-packet sending time is approximately equal to inter-packet arrival time. The rate adjustment of this approach is unsuitable for our system since it is proportional to the loss fraction of packets.

Considering the rate control mechanism followed by [53], probing the network is done at intervals, from which a smoothed available bandwidth average is calculated. This average, which is dependent on the round trip time (RTT) of the probing packets, is used as an upper threshold for the adjustable rate of flow. In our reasoning, this approach allows for the uneven distribution of network resources between clients. Taking into consideration two clients of same capabilities, but one further away from the server than the other, the more distant client has a lower threshold since it has a bigger RTT. Our system maintains an equal sharing of network resources under any circumstance.

In general, most of the literature found [1, 5] concerning rate control for multimedia streaming, except [53], regard the media streamed as MPEG video, thus assuming interdependency between frames as well as non-interaction. Our system's

requirements are unique since they demand a rate control for on-demand interactive streaming of independent images.

Chapter 3

Streaming System Design and Functionality

3.1 Streaming System Architecture

The system is based on the client/server architecture where the virtual environment is stored on the server side. On both server and client, there are several functional layers arranged from top to bottom. The highest is the application layer, which performs concrete features related to such applications as virtual shopping malls, remote medicine, or 3D games. The application layer is responsible for managing, compressing, and decompressing multimedia, and for performing the logic for an application scenario.

Below the application layer is the session layer. The session layer is responsible for managing and retaining states related to a streaming session. On the server side, each running session with a client has its own set of states and variables. It must respond to a client's request for a scene by fetching the corresponding images from the database. A running session must also keep track of each client's movement for the path prediction algorithm. Pre-fetching images is also done at this layer. These images are passed to the real-time protocol for streaming. On the client side, the session layer must manage the cache of recently acquired images. It must also translate the request for scenes from the application layer and pass it ton o the interactive streaming protocol. The client cache uses the Least Recently Used as a replacement policy.

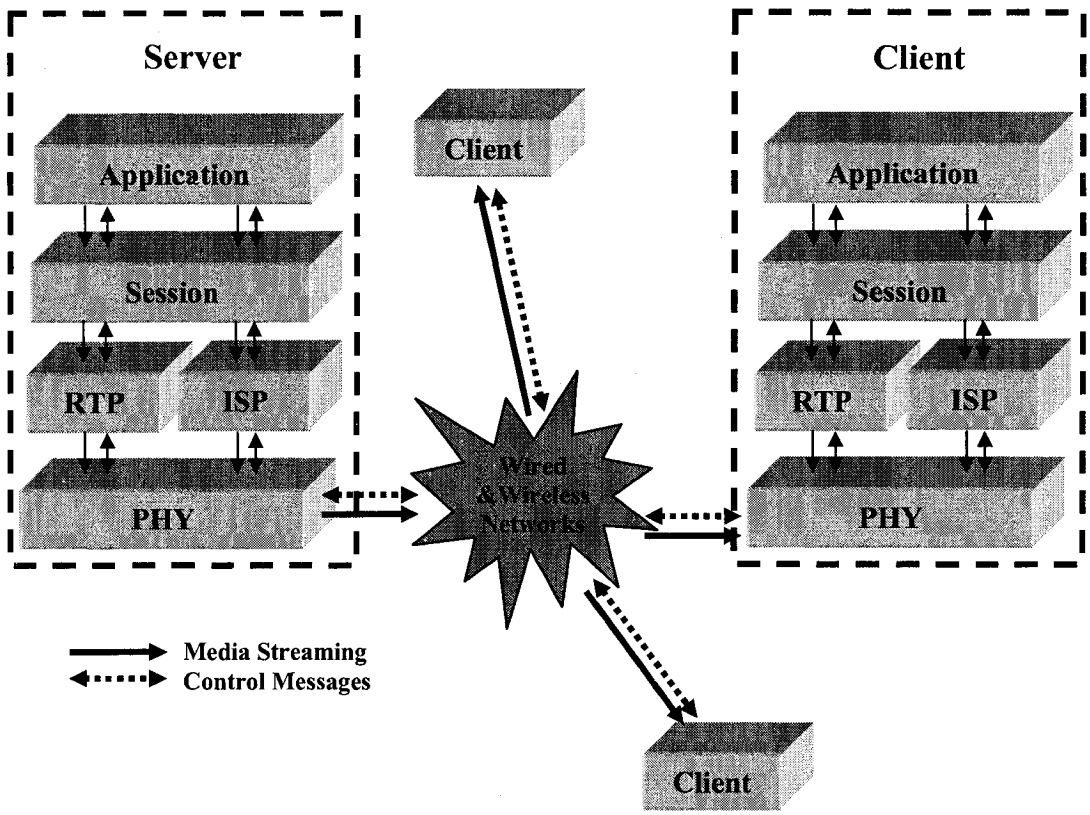

Figure 3.1: Streaming System Architecture

The ISP protocol controls multiple streaming sessions by choosing delivery mechanisms based upon Real-time Transport Protocol (RTP). It is the signaling protocol that accounts for the interactivity of the system. Among its tasks are message exchange for session initialization, image requests, and other control functions.

The RTP provides end-to-end network transport functions. It specifies the packetization scheme and a unique payload format since Morphing-JPEG is utilized as a compression algorithm. An immediate feedback mechanism is also integrated into the RTP protocol to constantly monitor network status. It provides a rate control algorithm with the necessary information to adjust the transmission rate.

3.2 Interaction of Functional Layers

When a mobile client wishes to navigate through a virtual environment stored on the server, the first task is setting up a streaming session. The application passes the connection request to the session layer, which in turn passes it to the ISP so as to send the appropriate connection request to the server. Upon receiving the request, the ISP on the server side will pass the request to the session layer, where a separate session is initialized with the proper session identification. The server's ISP will carry the session initialization arguments passed from the session layer to the client, where all necessary resources are allocated in terms of processing power and memory.

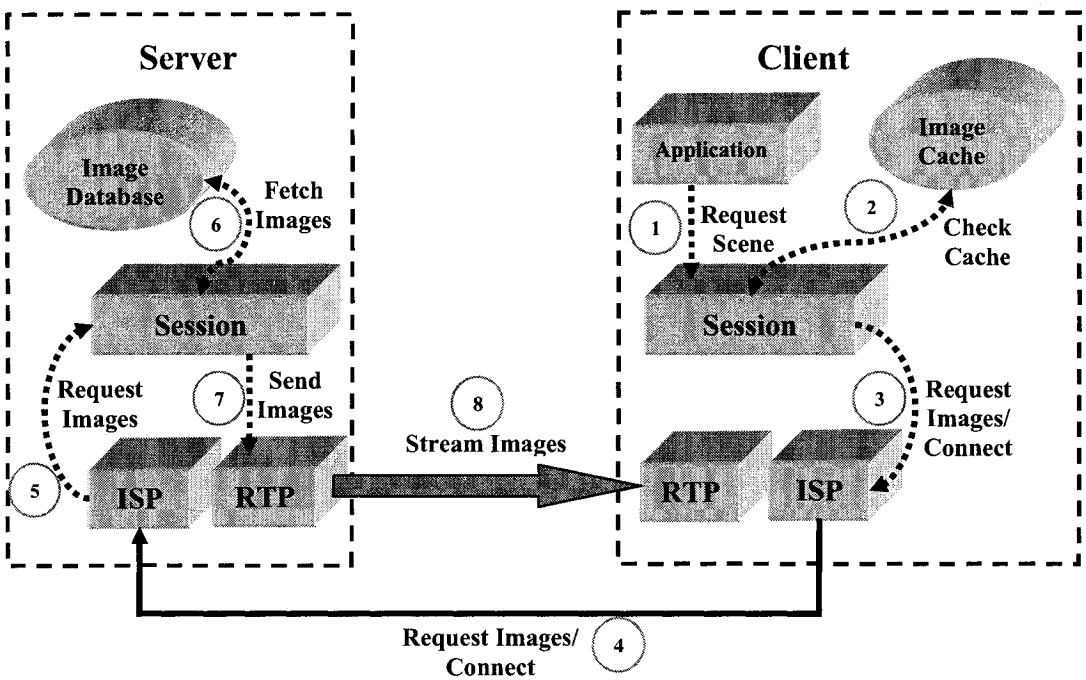

Figure 3.2: Functional Layer Interaction

As the client navigates through the virtual environment, it requests required scenes based on its position and orientation in the world coordinate system (*x, y, view direction*). The session layer will receive the request from the application and check its cache for the requested images. If they are not found, a request is sent to the server using the ISP with the *x, y, and view direction* parameters. Using these parameters, the server can identify which images are to be streamed to the requesting client. The server's ISP passes the parameters to the session layer, where fetching of the images is done. The images are then passed to the RTP to be streamed to the client. Upon arrival to the client's cache, a message is propagated across the session layer and the application layer to start the rendering process.

3.3 Session Layer

The session layer is directly below the application layer. It addresses issues such as establishing and terminating communications, fetching and pre-fetching images from the database on the server side, and deciding which images to request on the client side. Our proposed session protocol differs in implementation for each server and client.

Client Session Layer: The session layer on the client side initially establishes communication with the server. It does this by sending a request using the ISP. Once a client application has requested a 3D scene via its view parameters, the session layer of the streaming system processes the request. It checks the client's cache for an image of exact coordinates. If no image is found with exact coordinates, only the ones with matching view directions are considered. The session layer then requests any images closer to the required one than the ones the client has from the server via the ISP. For this

reason, the ISP request message contains the requested image as well as the closest image available in the cache. If no images with exact view direction are found in the client's cache, a request is sent to the server directly via the ISP. The following flow chart illustrates the algorithm:

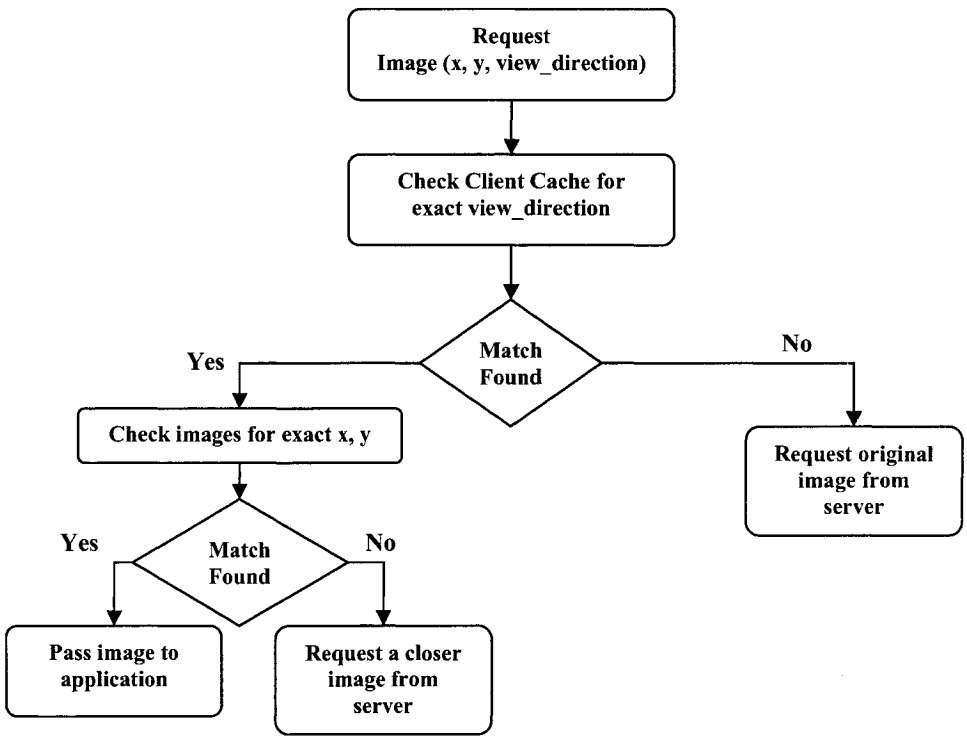

Figure 3.3: Client's Session.

Server Session Layer: Upon receiving a connection request from a client, the server creates a session to handle the client's requests. Once a client has been accepted, the session layer is responsible for fetching any image the client requests from its database. Requests for images are passed from clients to the server via the ISP and are then passed

onto the session layer. The request is then processed; the server searches the database for the two closest images to the requested one and returns them. If the request refers to an original image, those two images (one of them might be the exact match for the request) are passed to the RTP protocol for packetization and streaming. If the request is for closer images, then the images found in the database must be compared to those that the client has. They qualify for streaming only if they are closer. If a client wishes to disconnect from the server, it simply sends its request via the ISP.

The following flow chart illustrates the algorithm the session layer follows on the server side:

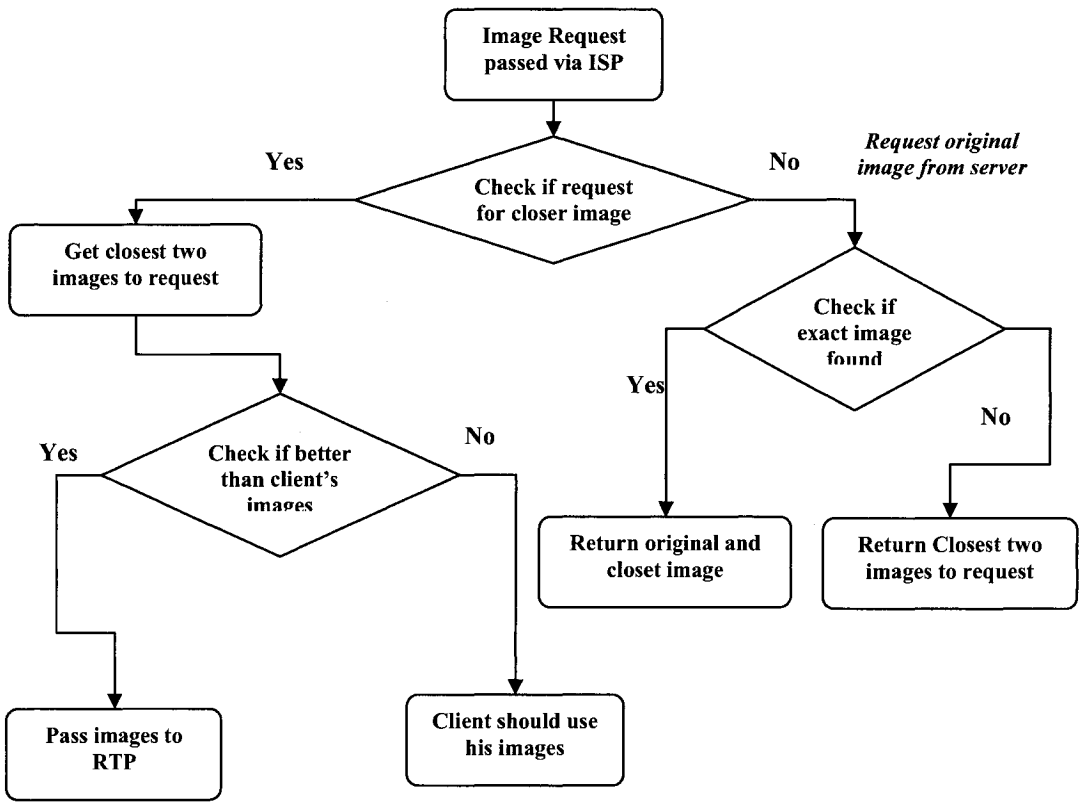

Figure 3.4: Server Session.

Each handling session also keeps track of the client's movement. For each request for images, the position of the client is recorded. Assuming the continuity of the client path, the server predicts the future position of the client and pre-fetches the corresponding images ahead of time. The set of pre-fetched images is arranged from closest to farthest from the client's current position and is passed to the RTP. If sufficient bandwidth is available, the RTP can selectively stream some of the pre-fetched images. This will save resources as well as maximize throughput.

3.4 Path Prediction and Pre-fetching Mechanism

The path prediction mechanism bases its calculations on the client's past and present motions. Depending on the path it took and the one it is currently taking, the prediction mechanism assumes with high probability that the client will continue on the same path. The first step that the path prediction mechanism undertakes is recording the client's path based on the requests it receives from the client. Whenever a client requests an image, meaning it is currently in the position that the image visualizes, the server records the coordinates of the client's position with respect to the world coordinate system.

The client takes on one of two possible modes when navigating within the virtual environment. His motion can either be linear or rotational. Thus, when navigating in the environment, the client either walks straight or turns around in his place. The server is kept aware of the client's navigation mode at all times and is informed of changes. Along with the client's previous positions, or path history, the navigation mode serves as a critical piece of data on which the path prediction mechanism can base its deductions.

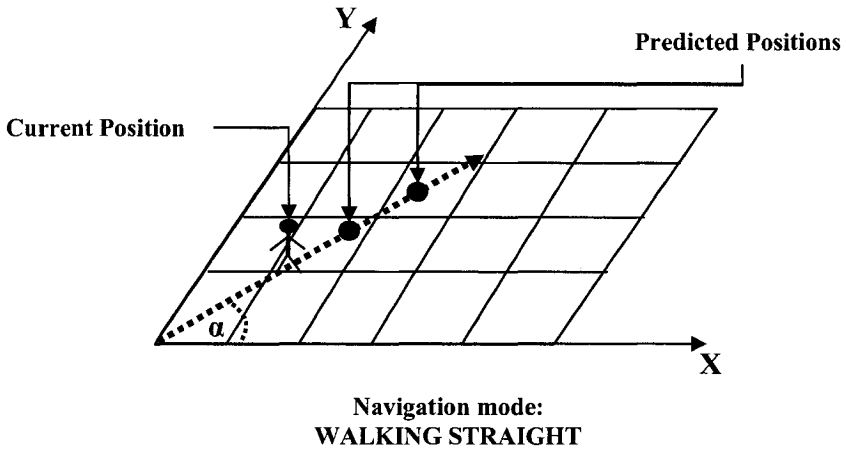

Figure 3.5: Prediction for linear motion

Considering walking to be the navigation mode, the view direction will remain constant whereas the X and Y coordinates will differ. The client will be simply walking in a straight line whose slope (or angle of orientation) is fixed. Thus, the path prediction mechanism assumes that the client will most probably continue on the same path. In this case, the predicted positions will be the ones that have the same view direction as the current and previous positions, but with different Xs and Ys. The predicted positions are arranged with those closer to the current position having higher priority. Of course, all maintain the same orientation and direction as the current position. In the turning-around navigation mode, the client remains fixed while the view direction changes. Examining the previous positions (view directions) of the client, the prediction mechanism is able to conclude whether the rotation is clockwise or counterclockwise. Thus, the predicted positions will have the same X and Y as the current position, but with view directions confirming to the orientation by which the client is currently rotating. The predicted

positions are also arranged, with the ones having a view direction closer the current position given higher priority.

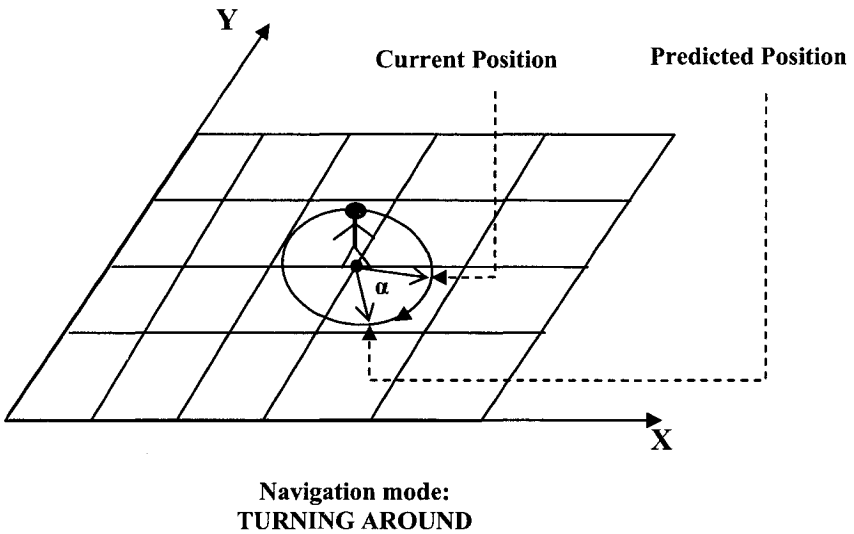

Figure 3.6: Prediction for rotational motion.

With the path prediction mechanism in place, the server can pre-fetch a set of images corresponding to the client's predicted positions. These images are then streamed to the client when ample bandwidth is available. The pre-fetching mechanism saves the client the trouble of sending a request and waiting for a response. Pre-fetching also maximizes the use of bandwidth and reduces the impairments set by the wireless network, making streaming run smoothly. Since path prediction relies on the client's current and previous history, an unused pre-fetched image is the one that is preceded by a change in navigation mode. Thus, the more changes in navigation mode, the less efficient pre-fetching becomes. Since the number of pre-fetched images is limited to two, a maximum of two images per change in navigation mode will not be used if they are

streamed in the first place. We also have to take into consideration the fact that the client must inform the server of any change in navigation mode before requesting any further images. This limits the number of unused pre-fetched images.

Another important point to consider about the path prediction and pre-fetching mechanism is that it has the ability to make decisions based on its awareness of the actual virtual environment. The mechanism might also use a probabilistic approach when picking pre-fetched images. In other words, it keeps track of which images are most frequently requested by clients and bases its prediction on this information.

3.5 Formal Implementation of Path Prediction and Pre-fetching Mechanism

The following is the formal implementation of the path prediction and pre-fetching mechanism. We assume the arrival of an ISP request packet at the server; the following steps occur after the ISP layer parses the packet and passes the necessary information to the session layer.

Step A0. Record X, Y, and view direction into list of previous positions.

Step A1. Retrieve last two positions in list $\{P_i (X_i, Y_i, \text{ViewDirection}_i), P_{i-1} (X_{i-1}, Y_{i-1}, \text{ViewDirection}_{i-1})\}$

Step A2. **If** (navigation mode → walking) **then**

If $(\text{ViewDirection}_i - \text{ViewDirection}_{i-1} \neq 0)$ **then**

No pre-fetching: *Insufficient information*

Else

If $(0 < \text{ViewDirection}_i < 90)$ **OR** $(180 < \text{ViewDirection}_i < 270)$ **then**

If $(X_i > X_{i-1})$ **And** $(Y_i > Y_{i-1})$ **then**

Pre-fetch all images **where** $(X_{i+1} > X_i)$ **And** $(Y_{i+1} > Y_i)$

Else If $(X_i < X_{i-1})$ **And** $(Y_i < Y_{i-1})$ **then**

Pre-fetch all images **where** $(X_{i+1} < X_i)$ **And** $(Y_{i+1} < Y_i)$

If $(90 < \text{ViewDirection}_i < 180)$ **OR** $(270 < \text{ViewDirection}_i < 360)$ **then**

If $(X_i < X_{i-1})$ **And** $(Y_i > Y_{i-1})$ **then**

Pre-fetch all images **where** $(X_{i+1} < X_i)$ **And** $(Y_{i+1} > Y_i)$

Else If $(X_i > X_{i-1})$ **And** $(Y_i < Y_{i-1})$ **then**

Pre-fetch all images **where** $(X_{i+1} > X_i)$ **And** $(Y_{i+1} < Y_i)$

Step A3. **If** navigation mode → turning around **then**

If $(X_i - X_{i-1} \neq 0)$ **OR** $(Y_i - Y_{i-1} \neq 0)$

No pre-fetching: Insufficient information

Else

If $(\text{ViewDirection}_i - \text{ViewDirection}_{i-1} > 0)$ **then**

Pre-fetch all images **where** $(\text{ViewDirection}_{i+1} > \text{ViewDirection}_i)$

If $(\text{ViewDirection}_i - \text{ViewDirection}_{i-1} < 0)$ **then**

Pre-fetch all images **where** $(\text{ViewDirection}_{i+1} < \text{ViewDirection}_i)$

Step A4. Arrange all pre-fetched images in order from closest to the requested image to furthest

Chapter 4

The Interactive Streaming Protocol

The Interactive Streaming Protocol (ISP) is a signaling protocol developed for the purpose of exchanging interactive requests and responses between streaming servers and clients through controlling multiple streaming sessions of image-based 3D scenes. The client and server use the interactive streaming protocol to launch a request for an action, send out a response, or report a state. The ISP is responsible among other things for the following tasks: locating the streaming server, establishing one or several streaming sessions, requesting the fetch or pre-fetch of original images from the server, and the teardown of a streaming session. The ISP protocol does not deliver the stream of 3D scenes itself but provides the interaction necessary for the Real-Time Transport Protocol (RTP) to fulfill this task.

When a client wishes to navigate through the virtual environment, it initializes a streaming session with the server through an ISP request for SETUP. The server, in turn, assigns enough resources, such as transmission buffers and CPU processing time, to the streaming session and delivers the general information of 3D scenes to the client. After receiving a response from the server confirming the establishment of a streaming session, the client chooses a 3D scene and navigation mode. The server then sends the corresponding world coordinate system and certain initial original images to the client. At this point, the streaming session is up, and the client can start its navigation in the virtual environment. During its navigation, the client sends ISP FETCH messages with its view parameters to request some original images. The server responds with the corresponding

original images. The server also tries to predict the client's walking path and pre-fetch a number of original images. If ample bandwidth is available, the server transmits these pre-fetched images in advance, without the client actually requesting them. The client selects a navigation mode at any time and informs the server of its navigation mode by a SET_NAVI_MOD message. Finally, when the client wants to end the navigation, it prompts the server to tear down the streaming session (TEAR_DOWN message) and release all corresponding resources.

The streaming system employs two types of transmission modes: *unicast* and *multicast*. To use unicast communication, the client chooses a port number and adds it to the request SETUP message. It is a usual application of the streaming system. To employ the multicast, the server selects the multicast address and port.

The ISP protocol employs UDP, which has been proved to be very efficient for real-time communication. In order to obtain the reliable signaling, the ISP protocol uses timers and a retransmission mechanism.

Our proposed ISP protocol is able to easily incorporate security schemes, especially the integration of web security mechanisms. The ISP can directly apply all HTTP authentication mechanisms such as basic and digest authentication. At the same time, the ISP protocol is friendly to proxy and firewall. Through using the same mechanisms as SIP (Session Initiation Protocol) and by adding *Record-route* and *Routes* headers, the ISP can be easily handled by both application and transport-layer firewalls.

In order to be consistent with current HTML internationalization efforts, the ISP protocol is defined to use UTF-8. Like SIP (Session Initiation Protocol), HTTP (Hyper Text Transport Protocol), RTSP (Real-Time Streaming Protocol), and SMTP (Simple

Mail Transport Protocol), our proposed ISP protocol is also plain/text so that it can be easily understood. It borrows a number of schemes from other widely used Internet protocols. The client-server design and the use of uniform resource locators (URLs) are elements of HTTP. The ISP also uses a text-encoding scheme and header style which are concepts of SMTP.

The ISP protocol contains three categories of messages: request, response and status. Both the server and the client can deliver a variety of messages. A request message is used to request an action from a peer; for example, a client can request to build a streaming session with the server. A response message is employed to respond to a request; for example, the server can respond to the client's request with an acknowledgment message. A status message reports a certain state. An ISP message consists of three parts: a start line, message headers, and a message body. The message body is optional and might use the Session Description Protocol (SDP). The start line is made up of a message type or a state number, ISP_URL, and the ISP version number. Message headers contain one or more headers depending on the type of message. The message body may be an SDP description.

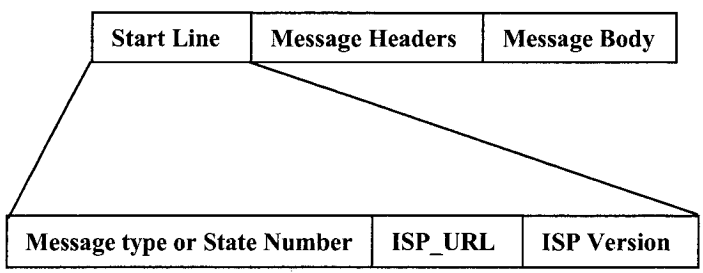

Figure 4.1: ISP message format

4.1 ISP Parameters:

Protocol parameters of the ISP protocol are included in the ISP start line. The ISP start line begins with either a message type or a state number. The next section explores ISP message types and ISP states. In the ISP start line, a message name or a state number is followed by the two parameters: ISP version and ISP_URL. These parameters are fixed for all types of ISP messages.

- ***ISP Version:*** This parameter is used to inform the peer of which version of the protocol is employed. The peer interprets an ISP message in terms of the specific version. This mechanism is consistent with other Internet protocols such as HTTP, RTSP, and SIP. Because our proposed ISP protocol is an initial version, we set this parameter to *ISP/1.0*.

- ***ISP_URL:*** The ISP_URL defines the destination. Intermediate devices such as proxies and firewalls are able to forward the ISP messages in terms of the ISP_URL. If the ISP_URL is a host name and a domain, the originator or the proxy can obtain the destination IP address through looking up a DNS database or requesting a location server. As mentioned earlier, the ISP protocol can use the unicast or multicast transport. The ISP_URL may specify a unicast address or a multicast address.

- ***FROM_URL***: Same as the ISP_URL but specifies the source.

4.2 ISP Headers:

An ISP message begins with a start line followed by one or more header fields. The header fields specify the integrated information of a protocol message. They maintain the ongoing streaming session, specify the navigation parameters, and describe a 3D scene. Different ISP headers are applied for different ISP messages.

- ***Session-ID***: The *Session-ID* is enclosed in each ISP message in order to specify the unique streaming session. In general, the navigation client creates a *Session-ID*. The *Session-ID* consists of two parts: a random set of characters and the client's hostname concatenated by the character '@'. The random characters ensure that the *Session-ID* is locally unique. The hostname guarantees the globally unique *Session-ID*. An example is: aFrtq123df@paradise.site.uottawa.ca.

- ***Command-Seq***: The *Command-Seq* header is mandatory for every ISP message. The ISP communication is round-trip, which means that an ISP message is sent and a corresponding acknowledgment is received. If a request or acknowledgment is delayed or lost, the entities can distinguish the relationship between requests and acknowledgments based on the *Command-Seq*. For example, *Command-Seq*: 2 FETCH is used by a FETCH message and an ACK method.

- ***View-point***: The *View-point* header stands for the coordinates of a view. It contains x and y coordinates of one or more viewpoints. The FETCH message sent by a client involves this header. A view vector in a 3D scene is associated

with each original image taken by a camera. A view vector is described by a viewpoint and a view direction. The viewpoint is specified by x, y, and z coordinates in the world coordinate system. The view direction is described by two components: an angle α between the projection of a view vector on the coordinate plane *X-O-Y* and x-axis, and another angle β between the view vector and the coordinate plane *X-O-Y*.

The view-point header can take three group coordinates which are (x, y, z), (x_0, y_0, z_0), and (x_1, y_1, z_1) at once. The first set of coordinates refers to the original image. The other sets refer to closer images.

As discussed earlier, we limit freedom of rotation in order to simplify our system. That is to say, the viewpoint moves only on a horizontal 2D plane parallel to *X-O-Y*. Hence, the viewpoint can be represented using x and y coordinates, and the view direction can be represented through the angle α between the view vector and x-axis. In the future, we may choose to extend the view vector's motion; the view-point header can handle these extensions.

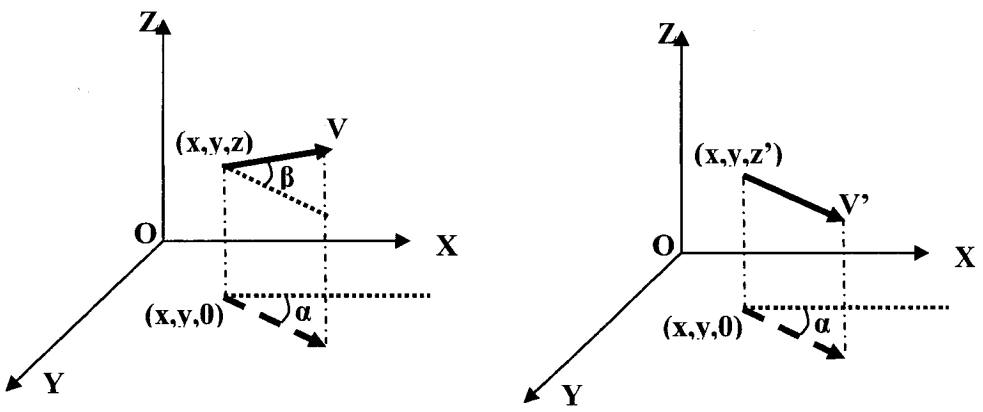

Figure 4.2: View Vector

- ***View-direction:*** The *View-direction* header contains the coordinates of a view direction. The view direction is described by two angles: α and β. We chose to simplify the system by choosing only α to describe the view direction.

- ***Navi-Mode:*** The SET_NAVI_MODE message uses this header. *Navi-Mode* has two possible values: looking around and walking. The looking around mode refers to rotational motion while walking refers to linear motion. Refer to section 3.4 for more details on the usage of navigation mode.

4.3 ISP Messages

The ISP protocol contains three categories of messages: request, response, and status. Depending on the message type, different headers are employed. Each message has a different functionality depending on which category it belongs to. A request

message is used to request an action from a peer, a response message is employed to respond to a request, and a status message reports a certain state.

4.3.1 Status Messages

The ISP protocol has some state messages used to report information to the peer. The server or client takes the corresponding actions according to these state messages. The state message employs the same syntax as HTTP and SIP messages: *a state number and the reason.* The ISP protocol borrows three state numbers from SIP. The 200 state stands for success. The 400 state represents an error on the client side. The 500 state means that an error has happened on the server side.

- ***"200 ok":*** The "200 state" is normally used to respond to the FETCH message. When a client requests some original images from the server for the current rendering image, the server responds with the *"200 ok"* message. The Real time Transport Protocol then transmits these original images. The client prepares for receiving the streaming once this state message arrives.

- ***"400 Client Error":*** The "400 state" is used to indicate some error related to the client. For example, an ISP request misses certain required headers such as *Session-ID* and *Command-Seq*. In this case, the state number is possibly followed by a specific error, for example "*Bad method*" instead of "*Client error*". When the client receives the "400 state" message, it corrects the error and resends the ISP message.

- ***"500 Server Error":*** The "500 state" is employed to represent an error caused by the server. For example, the server is unable to process a request because it is not supported. In this case, a "*Not Supported*" reason replaces "*Server error*". The client therefore understands what has happened, and takes corresponding action.

4.3.2 Request Messages

The Interactive Streaming Protocol employs a series of messages to request a specific action from its peer. Both a server and client are able to send ISP messages. Below we will discuss the different types of ISP request messages and their functions.

- ***SETUP***: The SETUP message is used to establish a streaming session between a navigating client and a server, and is generally originated by the client. The SETUP message involves the ISP_URL which specifies the location of the 3D data. If the server accepts the SETUP request, an acknowledgment message (*ACK*) is sent. If the server rejects the SETUP request, an error consisting of a number and a reason, like 400 client error or 500 server error is sent.

- ***SET_NAVI_MODE:*** A client uses a SET_NAVIGATION_MODE message to indicate the navigation mode currently in effect to the server. Two types of navigation modes exits: looking around and walking. The walking mode is the default navigation mode. Whenever the client investigates the scene, it can initiate the SET_NAVI_MODE message in order to change the navigation mode. The specified navigation mode has two functions. On one hand, it helps the server

predict the client's navigation path; on the other hand, it helps the rendering algorithm work in accordance with the client's intentions.

- ***FETCH:*** When the streaming session is in progress, the client sends FETCH messages to the streaming server to request one or more original images for the current rendering. If the client has in its cache images close to its current position in the world coordinate system, it informs the server of their view parameters in the FETCH message. The server decides whether or not certain original images will be sent or not by checking if better original images are found in its image collection. The v*iew-point* and v*iew-direction* headers are required in the FETCH message. The *view-direction* header supplies the client's current view direction. The *view-point* header takes the client's current view point and the nearest original images cached on the client whose view direction is same as the client's. The view-point coordinates of two cached original images are given first. The coordinates of the client's current view point are then listed. As mentioned earlier, *view-point* moves at the fixed level, which means that the z coordinate does not change in the world coordinate system. Therefore, all z coordinates are ignored, which means that they all are equal to the last view point of a client.

- ***PRE_FETCH:*** In some cases, the client is idle and is not requesting any original images. The PRE_FETCH message can be initiated by the server to make use of the available bandwidth. Some original images can be streamed without the client requesting them. The server memorizes the client's moving path. When ample

communication bandwidth is available, the server pre-fetches some original images and sends them to the client by predicting the client's navigation path from the recent route and navigation mode. The fluctuation of communication bandwidth seriously affects the smoothness of interactive streaming. If certain original images are pre-fetched, interactive streaming is not heavily impaired.

- ***SCENE INIT***: After establishing the streaming session, the server sends a SCENE_INIT message to initialize the navigation parameters. An image-based 3D scene is described using a large number of images that are located by the world coordinate and camera coordinate systems. When a client navigates in a scene, a virtual camera coordinate system is set to represent the client. The SCENE_INIT message delivers the initial position of the virtual camera coordinate system to the client. Thus, the SCENE_INIT message contains view-point and view-direction headers to mark the client's initial position in the virtual environment and describe the relationship between the world coordinate and virtual camera coordinate systems. The SCENE_INIT message is also used when the client leaves one scene to go to another; for example, it can move from a room to the next. Different scenes require the resetting of the world coordinate system.

- ***TEAR_DOWN***: The client or the server terminates a streaming session using the TEAR_DOWN message. Once the streaming session is terminated, all related resources, such as memory and processing threads, are released.

4.3.2 Response Messages

A response message is employed to respond to a request message. At this point there is only one type of response message, which is the acknowledgement.

- *ACK:* Whenever a request is delivered, an ACK message is expected. If the request fails, an error response is received. A timer is set when a request is originated. If an acknowledgment does not arrive within the specified period, the request will be retransmitted. The ACK message has the same *Command-Seq* header as the corresponding request message. In the case of loss or delay, the originator can easily understand the relationship between the request and the *ACK* message.

4.4 Operating Scenarios for the ISP Protocol

In order to have a better understanding of the concepts presented so far with regards to the ISP protocol, we present a set of typical scenarios illustrating message exchange and provide a discussion.

4.4.1 Establishing a Streaming Session

In order to establish a streaming session, the client originates the SETUP round-trip message sequence consisting of SETUP, ACK and "200 ok". In each message, the start line contains the ISP_URL of the streaming server, which is the destination address. The start line is followed by a FROM_URL header, which specifies the origin address. The unique *Session-ID* for each session is created by the client. The *Session-ID* is

adopted during the entire session. The *Command-Seq* header includes a number and a request name. This number linearly increases during the session between the different round-trip messages. The server and client make sure that the streaming session is established successfully using an ACK message from the server to the client and a "200 ok" state message from the client to the server.

After the streaming session has been created, the server delivers a SCENE_INIT message right away to initialize the navigation scene of the client. The SCENE_INIT message contains *view-point* and *view-direction* headers, which specify the initial position of the virtual camera coordinate system. That is to say, the client's initial view point and view direction. After delivering the ACK message for the SCENE_INIT message, the client begins to navigate the scene.

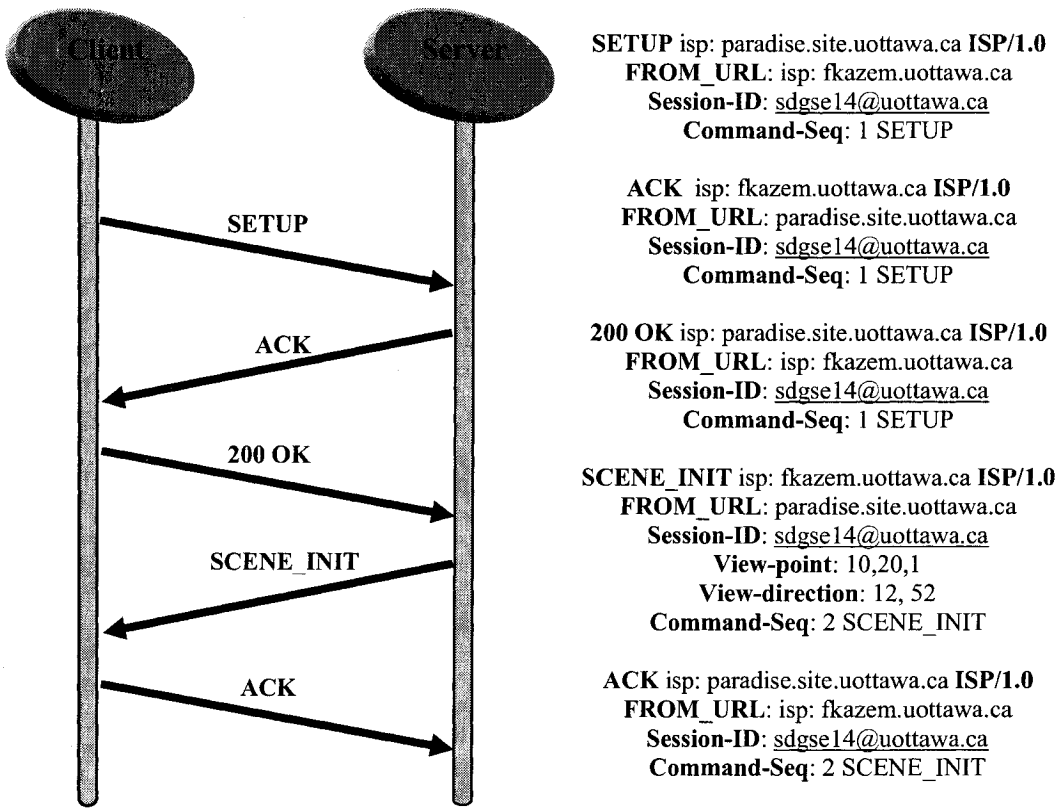

Figure 4.3: Establishing a Streaming Session

4.4.2 Fetching and Pre-fetching certain Original Images

During navigation, the client requests certain original images from the streaming server using the FETCH message in order to render the current image. The server first responds with a "200 ok" state message, then transmits the requested original images through the Real-time Transport Protocol (RTP). The *Session-ID* of the FETCH message and the "200 ok" state are identical to the one in the SETUP message includes. Both the FETCH message and the corresponding "200 ok" state employ the same *Command-Seq*. Once the client receives the "200 ok" message, it prepares to receive the RTP stream. The "200 ok" state message helps the client assign the necessary resources for RTP streaming in advance.

As seen in the above discussion, the PRE_FETCH message is employed to transmit some original images in advance. These original images are not requested by the client but sent by the server using its own initiative. The server analyzes the client's recent navigation path and current navigation mode to predict the future navigation path. The PRE_FETCH message does not have the view-point or the view-direction headers. The client responds with the "200 ok" state message to indicate that the following original images will be cached by the client.

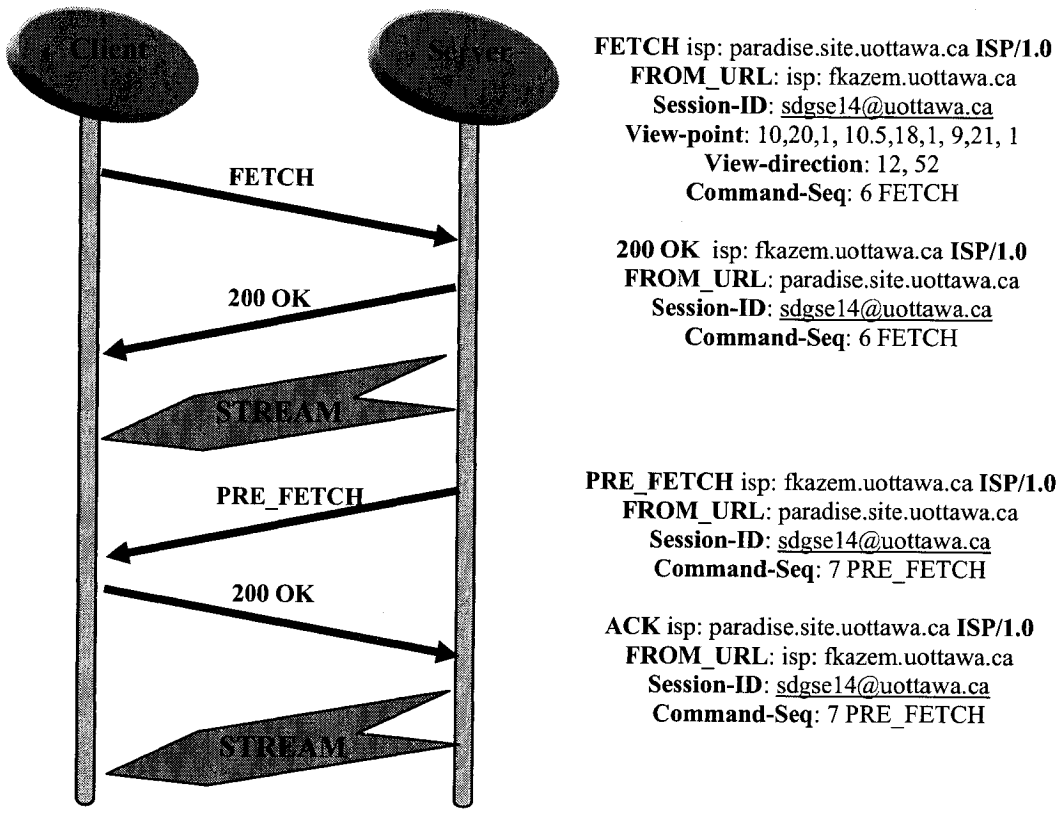

Figure 4.4: Fetching and Pre-fetching Original Images

4.4.3 Leaving to the Next Scene

In order to locate objects, independent world coordinate systems are built for different scenes. When the client walks to the joint boundary between two scenes, the server delivers a SCENE_INIT message. The client replies with a "200 ok" state message to indicate that a new world coordinate system has been set and that its position in the virtual coordinate system has been initialized. After arriving in the new scene, a SET_NAVI_MODE method may be sent out by the client. If the client does not deliver the SET_NAVI_MODE message, the server continues using the latest navigation mode.

4.4.4 Terminating a Session

The server or client can terminate a session. In the latter case, the client will send a TEAR_DOWN message, to which the server replies with a "200 ok" state message if it is able to terminate. The client finally sends an ACK.

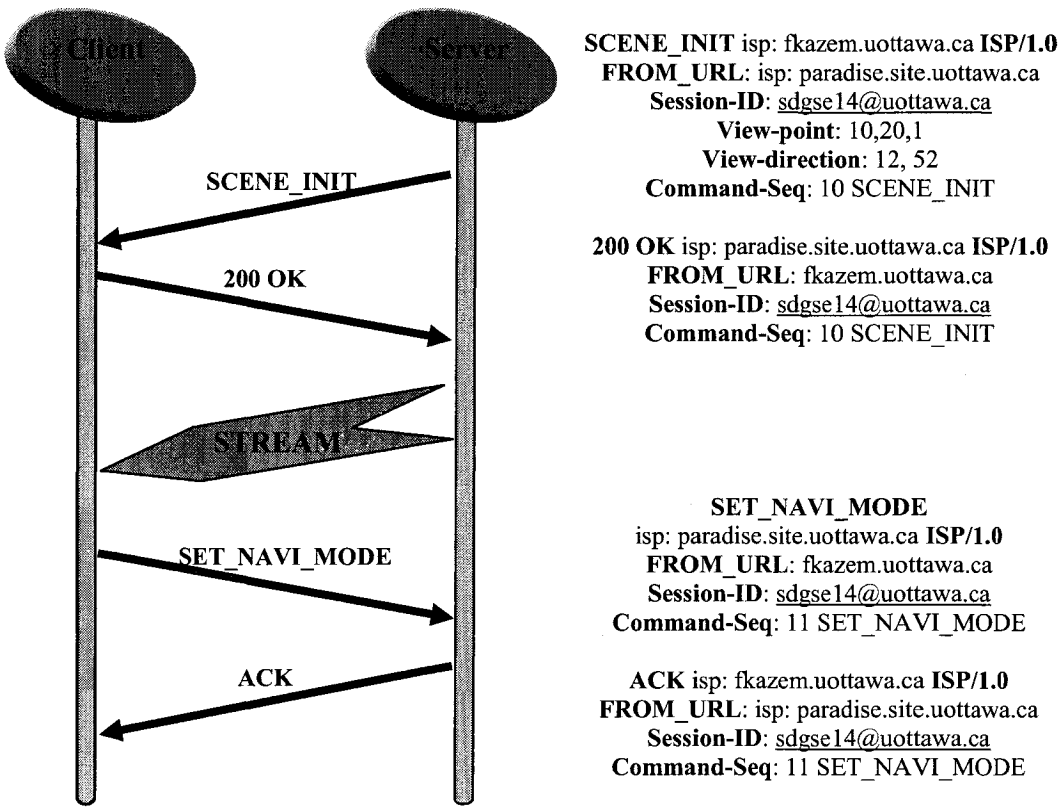

Figure 4.5: Leaving a scene

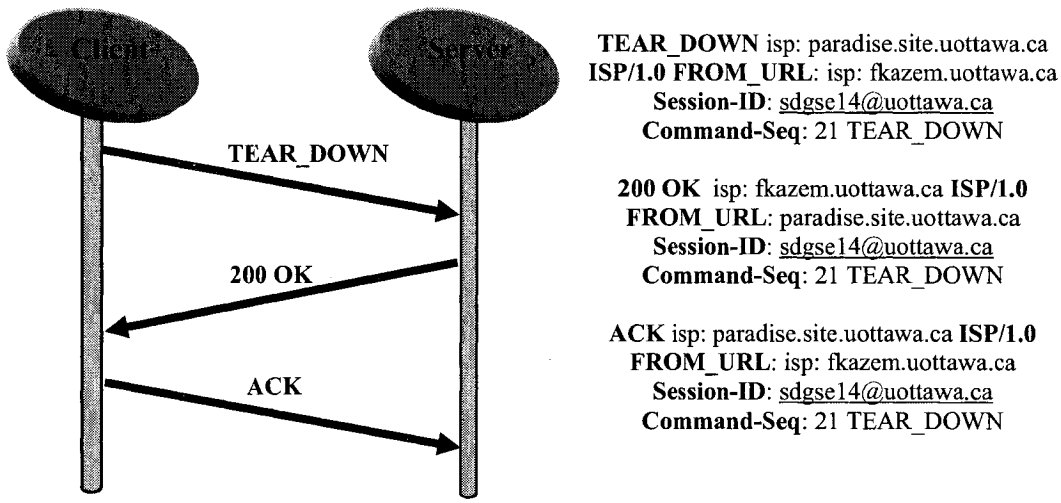

Figure 4.6: Terminating a Session

4.5 Internal States of Operation for the ISP Protocol

Below are the state diagrams for the operation of the ISP. For both the server and client, the ISP changes between three main states: SETUP, PROCESS, and TEAR DOWN. The SETUP state denotes the messages exchanged in order to establish a streaming session. The PROCESS state represents the normal functioning of the ISP where it processes requests for fetching, pre-fetching, etc. The final state is the TEAR_DOWN, in which the client and server interchange the proper messages for proper termination of the streaming session.

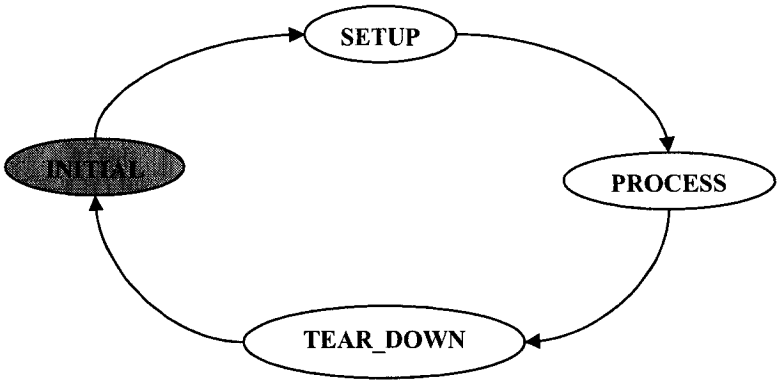

Figure 4.7: State Diagram for ISP

The implementation of each state varies between the server and the client. For example, the SETUP state of the client comprises sending the SETUP message and waiting for the ACK and the SCENE_INIT messages. While on the server side, it comprises sending the ACK and SCENE_INIT upon arrival of the SETUP message. The implementation of a state does not only include the proper message exchange. Any necessary processing is done in the state as well. For example, the implementation of the FETCH state on the server in the diagram below includes passing scene parameters to the session layer.

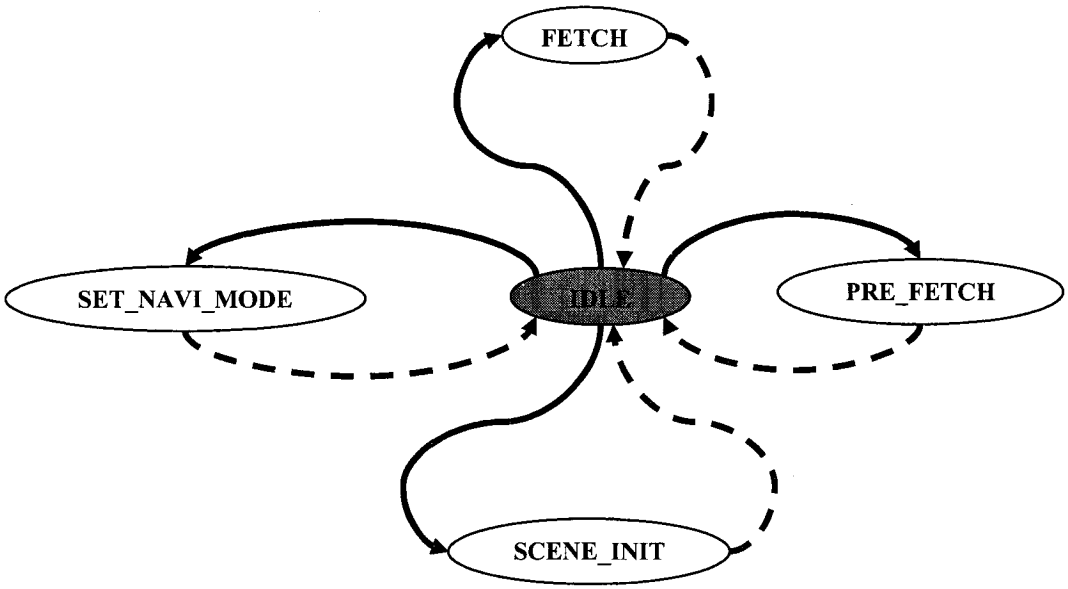

Figure 4.8: Internal State Diagram for the PROCESS State

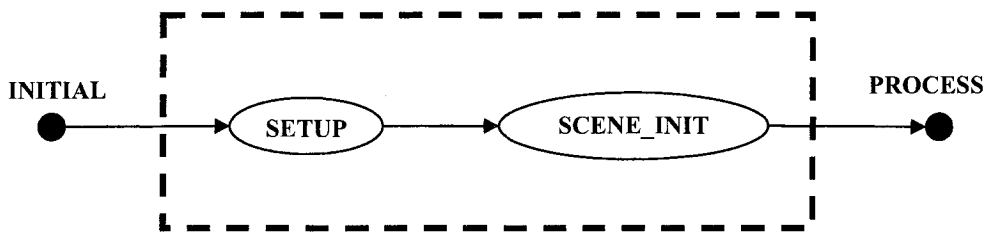

Figure 4.9: Internal State Diagram for the SETUP State

Chapter 5

Real Time Transport Protocol

When a mobile client connects to the server, it requests the required scenes based on its position and orientation in the world coordinate system (*x, y, view direction*). Using these parameters, the server can identify which images are to be streamed to the requesting client. As the client navigates through the virtual environment, its position with respect to the world coordinate system will change, thereby requiring a new image to visualize its current view. For every new position, the client requests the corresponding image from the server and stores it in a cache of recently received images. Thus, the streaming process is on-demand; it is relative to the client's navigation in the virtual environment. In other words, the server streams images only when it receives requests from the client, who only issues these requests when there is motion within the virtual environment.

In video streaming for example, the server is aware beforehand of the amount of data to be streamed and knows that the stream is continuous. Unlike video streaming, our interactive streaming system remains unaware of the amount of data required for streaming and the time at which streaming is requested. This interactive discontinuous streaming processm, which runs in the limited resource wireless network, leaves little or no space for retransmissions and buffering. Under these unique restrictions, new schemes are integrated into our streaming protocol to render it more efficient.

Taking into consideration the restrictions pertaining to wireless networks and the nature of the streaming process discussed earlier, we propose some schemes to meet the

system's requirements. An immediate feedback mechanism provides the rate control algorithm with sufficient data to continuously monitor the network status so that its rate can be adjusted accordingly. With its constant monitoring, the rate control algorithm is able to minimize the loss rate by unburdening the network when bandwidth is limited and by maximizing throughput when it is ample. The path prediction and pre-fetching mechanism introduced earlier serves as a means of saving time by predicting a client's future movement and streaming corresponding images when bandwidth is ample.

Finally, a unique payload format is required to account for the Morphing-JPEG compression algorithm, as well as a packetization scheme to enhance the robustness of the streaming.

5.1 RTP Payload Format for Image Morphing

The Real-time Transport Protocol (RTP) is used to carry payloads of the real-time application. It provides end-to-end delivery services for data with real-time characteristics, such as interactive audio and video. These delivery services include payload type identification, sequence numbering, time stamping, and delivery monitoring.

For the streaming of any multimedia compressed data over wired or wireless networks, a private payload format over RTP such as H.26x over RTP [32], [35] and G.7xx over RTP [27], [36] is required. The Morphing-JPEG compression algorithm has certain unique features and working mechanisms. For example, encoded streaming is transmitted in a video unit consisting of one or two images fetched from two separate positions. In addition, the Morphing-JPEG algorithm performs layer-transcoding, in which a different number of layers are transmitted in different cases, depending on the error-prone wireless

network. Therefore, the RTP payload format for the Morphing-JPEG is designed in terms of these features and requirements.

For each RTP packet, the RTP fixed header is followed by the Morphing- JPEG payload header, which is followed by the Morphing-JPEG compression codestream. The structures of the Morphing-JPEG payload header and Morphing-JPEG compression codestream are unique for the streaming of the Morphing-JPEG algorithm. The following figure depicts the structure of an RTP packet.

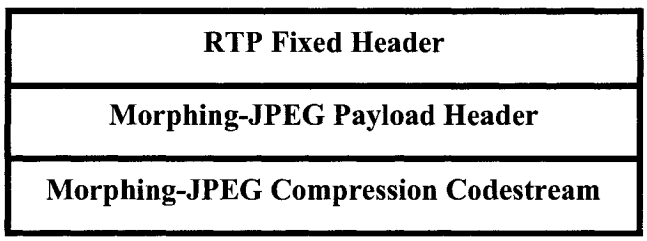

Figure 5.1: Structure of RTP packet for Morphing-JPEG

5.1.1 RTP Fixed Header

The fields for the RTP fixed header are the same as those of other RTP applications. Only the following fields are exceptions; they are specific to the Morphing-JPEG algorithm:

1. ***Payload Type:*** RTP standard describes that this field specifies the format of the RTP payload and determines its interpretation by the application [26]. Because our payload type is not specified by the RTP profile-that is to say, the payload

code is not assigned through RTP means-the upper layer defines the payload code.

2. ***Number of Video Unit***: When Morphing-JPEG encoded images are streamed over wireless networks, the transmission stream is constructed from a series of video units consisting of one or more images. In general, a video unit is divided into more than one fragment. *Number of Video Unit (NVU)* is a random value given to all RTP packets in a video unit. This field helps the receiver identify the complete transmission of a video unit. Unlike the conventional RTP protocol , which makes use of the marker bit of the last packet of a video, the *Number of Video Unit (NVU)* of all RTP packets of a video unit are the same, but one number higher than those of the previous video unit. Once the receiver detects the *NVU* is different from the previous one, it disposes of the last video unit. This approach works well with wireless communication where the last RTP packet may be lost.

3. ***Timestamp***: The Morphing-JPEG stream has no strict sampling instance. Unlike other multimedia, the timestamp for Morphing-JPEG codestream does not indicate the sampling instance. Nevertheless, it is significant for calculating synchronization and jitter when other media streams are associated with Morphing-JPEG. Hence, it is required to increase monotonically and linearly. All RTP packets for the same video unit will set the same timestamp.

5.1.2 The Morphing-JPEG Payload Header

The fields of the Morphing-JPEG payload header are depicted in the following figure, with a discussion given shortly afterwards:

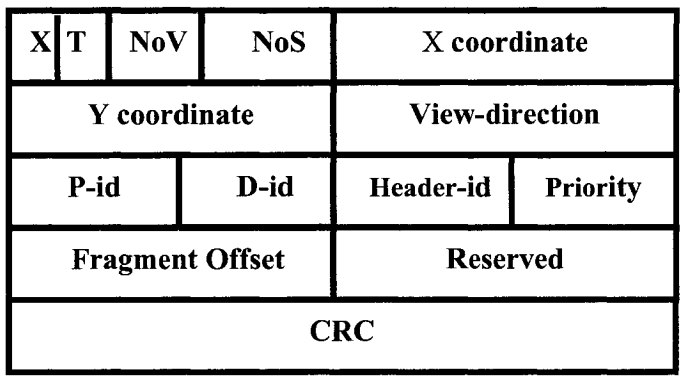

Figure 5.2: Fields of Morphing-JPEG Payload Header

1. ***Payload header extension bit flag (X):*** Indicates whether there is an optional Morphing-JPEG payload header following this payload header by setting the bit to 1 (0 otherwise). Each optional payload header should mark whether more optional payload header follows. This field facilitates the receiver's parsing of the RTP payload.

2. ***Twin images bit flag (T):*** This field indicates whether one or two original images are being transmitted or just one. For the Morphing-JPEG compression algorithm, the novel image for the current view is rendered either directly from an original image directly, or through morphing two original images of the image collection.

T bit is set to 0 when one original image is streamed, and is set to 1 when twin original images are transmitted in RTP packets.

3. ***Number of an image in a video unit (NoV):*** Since the client keeps a number of images in its cache, it is possible that the server will only need to transmit one or no images. *NoV* indicates the number of packed original images with RTP packets.

Table 5.1: NoV Field Definition

NoV **Value**	**Definition**
00	Only one transmitted image
10	The first one of twin original images
11	The second one of twin original images

4. ***Number of an image in the streaming (NoS)*:** Like with the *NVU,* the receiver should be able to detect when an image has been completely transmitted. The traditional solution is to set a marker bit to mark whether or not the image has been completely transmitted. In error-prone wireless communications, the last fragment of an image is may be lost. For *NoS,* the sender assigns a random number to the first image. For each image, the *NoS*s for all fragments are the same but are incremented by one for the next image.

5. ***Viewpoint and Direction of View (X coordinate, Y coordinate, view-direction):*** A world coordinate system is built in the 3D scenes where all objects, scenes, and views can be located. The X and Y coordinates of the viewpoint and the direction of view are associated with the corresponding images. In order to facilitate searching and locate a certain acquisition image in the encoded video stream, these camera parameters are presented in a plain-text manner. On the client side, these camera parameters are used to render a novel image.

6. ***Viewpoint and View-direction Identification Value (P-id and D-id):*** The server and the client have a common world coordinate system. The entire "walking field" is divided into a number of small sections. The whole view-direction (Omni-direction) is partitioned into many small equal angles. Each small view section and each small angle are then represented by an identifier respectively. Since Morphing-JPEG is transmitted over error-prone wireless networks, the fields *X coordinate*, *Y coordinate* and *view-direction* may arrive corrupted, which would affect the retrieval of the correct image. The two fields *P-id* and *D-id* represent a small view area and a small range of view-direction, and can provide an extremely approximate value in identifying the coordinates of the correct image. The fields *P-id* and *D-id* make streaming more robust.

7. ***Codestream Header Identification Value (header-id):*** The compression main header of the Morphing-JPEG compression algorithm contains certain necessary compression parameters such as a quantization step and a motion vector. If these

parameters are corrupted, the entire codestream cannot be decoded. In general, neighboring encoded images have the same compression main header, which is marked by a unique integer. The field *header-id* takes this integer. If the codestream main header is corrupted, the right codestream main header can be picked up according to the field *header-id* in the client's cache memory.

This mechanism is different from other multimedia videos such as JPEG2000. For JPEG2000 over RTP, there is a similar field *header-id.* However, it works in a different manner. If the current main header is different from the previous main header, the field *header-id* is increased by one. If the current main header is corrupted, the previous main header can be made use of by checking whether the field *header-id* is same as the previous one. In JPEG2000, this principle is based on the continuous codestream.

8. ***Priority of Each Image Layer:*** The Morphing-JPEG algorithm compresses an image into a basic layer (lowest layer) and several enhanced layers. It is important to include the priority of the layer with the packet in order to guarantee its proper transmission.

9. ***Fragment Offset:*** Each compressed image is generally bigger than the capacity of an RTP packet. Therefore, the compressed codestream of an image may be fragmented into several packets. This field facilitates the reassembling process for the client.

10. ***Reserved Bits for the Morphing-JPEG Algorithm:*** Because the streaming system may be adapted to different applications, more information may be required to transmit in the RTP packet. This field is preserved for the future.

11. ***Cyclical Redundancy Check (CRC)***: Since wireless communication is error-prone, Cyclical Redundancy Check (CRC) is necessary. CRC is used to detect whether or not the payload header is corrupted and, if it is, to try to correct corrupted bits.

Table 5.2: Fields of Morphing-JPEG Payload Header

Fields	Bits	Simple Description
X	1	Payload header extension flag
T	1	Twin images bit flag
NoV	2	Number of an image in a video unit
NoS	12	Assigned number of an image in the streaming
X coordinate	16	X coordinate of viewpoint for the original image
Y coordinate	16	Y coordinate of viewpoint for the original image
View-direction	16	The view-direction of the current image
P-id	8	Viewpoint identification value
D-id	8	View-direction identification value
Priority	8	Priority for each layer of image
Header-id	8	Codestream header identification value
Fragment Offset	16	The byte offset in the image
Reserved	16	Reserved bits for compression algorithm
CRC	32	Cyclical redundancy check for payload header

5.2 Packetization Scheme for View Morphing of Image-based 3D Scenes

A Morphing-JPEG codestream is composed of a header and a main section. The header section is the main header beginning with the start of the codestream (SOC) marker. The main section includes one or more layers of an image. One image can be compressed into one basic layer or several enhanced layers because of progressive resolutions, which Morphing-JPEG supports. Finally, the end of the codestream (EOC) marker indicates the end of the codestream.

The RTP server divides the encoded stream of one image into several small pieces, each included in an RTP packet. In the header of the RTP packet, the *offset* field represents the position of the current piece of the encoded stream. Because the wireless network is error-prone, it is better to have these pieces relatively independent. Thus, the concept of *packetization item* is introduced; this is defined as a component of the Morphing-JPEG codestream such as a main header, a layer header, or a pixel block of the compressed codestream.

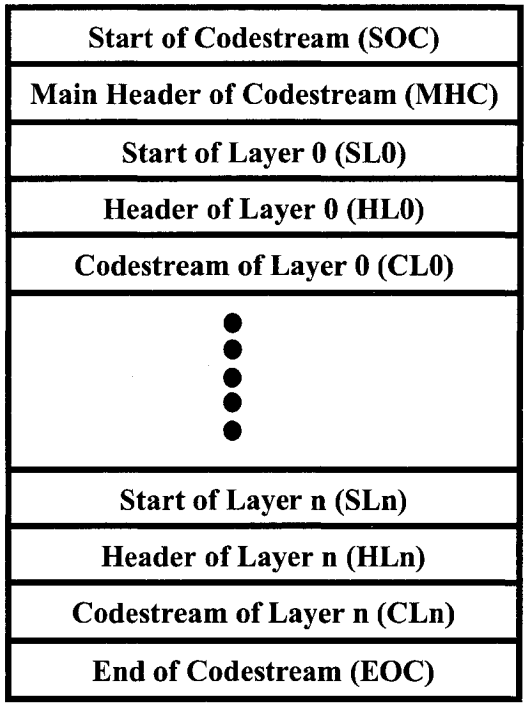

Figure 5.3: Structure of Morphing-JPEG codestream

The server partitions the Morphing-JPEG codestream into multiple packetization items by parsing the codestream, then packs these packetization items into RTP packets. An arbitrary number of packetization items can be packed into an RTP packet in the correct codestream order. Additional padding octets are required because a packetization item may not be 32-bits aligned. However, any required padding must be added at the end of the packetization item. If a packetization item is too large, it can be fragmented.

RTP Fixed Header	Morphing-JPEG Payload	Packetization Item 0	Packetization Item 1	Packetization Item 2

Figure 5.4: A Packetization scheme with multiple packetization items

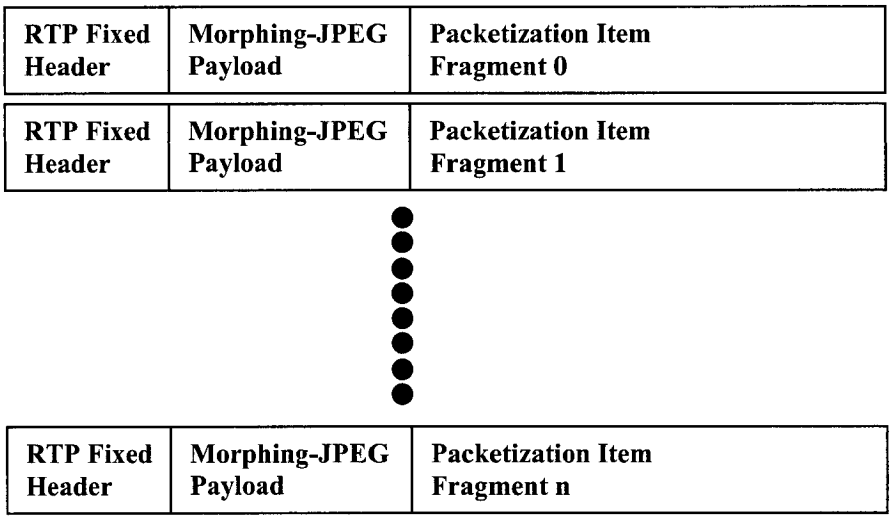

Figure 5.5: A Packetization scheme with fragmented packetization item

5.3 Formal Implementation of Payload Format and Packetization Scheme

We now formally describe the implementation for our proposed RTP payload format and packetization scheme.

*A. **Implementation on the server***

We have two main processing threads. The first parses the Morphing-JPEG compression codestream into one or more packetization items or packetization item fragments. The second constructs the RTP packet from the RTP fixed header, the Morphing-JPEG payload header, and the packetization codestream parsed by the first procedure.

*1) **The first process thread:*** Parse the Morphing-JPEG codestream of a video unit

Step A0. Parse the Morphing-JPEG compression codestream of a video unit into a number of packetizaiton items.

Step A1. According to the order of the packetizaton items created by the step A0, calculate the number of Bytes of each packetization item. If the packetization item is not 32 bits aligned, the end octets of packetization item are padded with 0.

If the length of packetization item > 1500Bytes (Notice: Adjusted by experiments) **then**

Partition the packetization item into several packetization item fragments in the unit of 1500.

Step A2. The variable *PacketizationData* installed in the RTP packet ← 0.

Step A3. Add a packetization item or a packetizaiton item fragment to the variable *PacketizationData.*

Step A4. If the length of the variable *PacketizationData* + the length of the next packetization item < 1500 **then**

go to Step A3.

Step A5. Invoke the second processing thread, construct the full RTP packet and send it to the network layer.

Step A6. If the video unit is not transmitted completely **then**

go to Step A2.

else

go to Step A0.

*2) **The second processing thread:*** Construct an RTP packet from an RTP fixed header, Morphing-JPEG payload header, and packetization codestream

Step B0. Set the field *payload type* for the Morphing-JPEG in the RTP fixed header through searching the definition of the upper layer.

Step B1. If the current packet is the first fragment of a video unit or the current packet is a whole video unit **then**

If this packet is the first one of RTP session **then**

the field *NVU* in the RTP fixed header ← *a random value*

else

the field *NVU* in the RTP fixed header ←*the last NVU* +1;

else

the field *NVU* in the RTP fixed header ←*the last NVU.*

Step B2. If the current packet is the first RTP packet of the communication session **then**

The field *Timpstamp* in the RTP fixed header initialized to a random value;

else If the current packet is the first RTP packet of a video unit or the current packet is a whole video unit **then**

The field *Timestamp* in the RTP fixed header ←last *Timestamp* + 1/90K (90K clock);

else

The field *Timestamp* in the RTP fixed header ←last *Timestamp.*

Step B3. If an optional payload header follows the Morphing-JPEG payload header **then**

The field *X* in the Morphing-JPEG payload header ← 1;

else

The field *X* in the Morphing-JPEG payload header ← 0.

Step B4. If the current packet is the first fragment of an image or the current packet is the whole image **then**

If this packet is the first one of RTP session **then**

The field *NoS* in the Morphing-JPEG payload header ← *a random value*

else

The field *NoS* in the Morphing-JPEG payload header ←*the last NoS* +1;

else

The field *NoS* in the Morphing-JPEG payload header ← *the last NoS*.

Step B5. If the current rendered image is an original image **then**

The field *T* in the Morphing-JPEG payload header ← 0;

else

The field *T* in the Morphing-JPEG payload header ← 1;

Step B6. If only one original image is transmitted for the current rendered image **then**

The field *NoV* in the Morphing-JPEG payload header ←00;

else If this is the first original image for the current rendered image **then**

The field *NoV* in the Morphing-JPEG payload header ← 10;

else If this is the second original image for the current rendered image **then**

The field *NoV* in the Morphing-JPEG payload header ← 11;

Step B7. Set the fields *X coordinate*, *Y coordinate* and *view-direction* in term of the application payload.

Step B8. Set the field *P-id* and *D-id* in the Morphing-JPEG payload header through looking up the coordinate system in light of the position of view-point and the direction of view.

Step B9. Set the field *header-id* in the Morphing-JPEG payload header through searching for the codestream header identification value in a codestream header table.

Step B10. Set the field *priority* in the Morphing-JPEG payload header through looking up the priority table.

Step B11. If the packet is the first RTP packet of an image **then**

The field *offset* in the Morphing-JPEG ← 0;

else

offset ←last *offset* + the length of the Morphing-JPEG compression codestream carried by last RTP packet.

Step B12. All bits of the field *reserved* in the Morphing-JPEG payload header are set to 0.

Step B13. Construct a RTP packet from RTP fixed header, Morphing-JPEG payload header and the variable *PacketizationData* created by the first processing thread.

Step B14. Transfer the RTP packet to the network layer.

*B. **Implementation on the client***

Parse the RTP packet, send packetization items or packetization item fragments to the application layer and indicate that the application layer deal with the Morphing-JPEG codestream.

Step C0. Parse the RTP Fixed header.

If the field *payload type* is for Morphing-JPEG **then**

go to Step C1.

else

go to the implementation for other RTP algorithms.

Step C1. If the field *NVU* in the RTP fixed header is different from the last one **then**

Instruct the application layer to render the current image at the moment.

Step C2. Pick up the field *Timestamp* from the RTP fixed header and Transfer it to the application layer.

The application process thread uses it to calculate the synchronization and jitter.

Step C3. If the field *X* in the Morphing-JPEG payload header is 1 **then**

Locate all Morphing-JPEG optional payload headers, parse the packetization data and transfer the packetization data to the application process thread.

else

Directly parse the packetization data and transfer the packetization data to the application process thread.

Step C4. If the field *NoS* in the Morphing-JPEG payload header is different from the last one **then**

go to Step C5.

else

go to Step C7 .

Step C5. If the field *NoV* in the Morphing-JPEG is 00 **then**

Inform the application process thread to decode the codestream of the original image.

If the field *T* in the Morphing-JPEG payload header is 0 **then**

Instruct the application process thread to render the current image directly using this original image;

else

Instruct the application process thread to render the current image through morphing this original image and another image fetched from the client's local memory.

else If the field *NoV* in the Morphing-JPEG is 10 **then**

Instruct the application process thread to decode the codestream of the current original image

else

Instruct the application process thread to decode the codestream of the current original image. And inform the application process thread to render the current image through morphing this original image and the previous original image.

Step C6. Implement the CRC algorithm to check whether the position of viewpoint and the direction of view are correct.

If they are correct **then**

go to Step C7.

else If the position of viewpoint is wrong **then**

Run the CRC algorithm to make sure the field *P-id* is correct, then get the approximate position of viewpoint according to the *P-id.*

else If the direction of view is wrong **then**

Run the CRC algorithm to make sure the field *D-id* is correct, then get the approximate direction of view according to the *D-id.*

else

Abort this current transmitted original image and inform the application process thread.

Step C7. Check the field *Priority* in the Morphing-JPEG payload header and instruct the application about the transcoding layer.

Step C8. Run the CRC algorithm to check whether the main header is correct.

If the main header is correct **then**

go to Step C9.

If the main header is wrong **then**

Run the CRC algorithm to check whether the *header-id* is wrong. If the *header-id* is correct, the right main header is selected according to the *header-id.*

Step C9. Skip the field *Reserved* in the Morphing-JPEG main header.

Step C10. Pick up the field *offset* in the Morphing-JPEG main header.

If the *offset* = last *offset* + the length of packetization data **then**

Append the packetization data to last packetization data;

else

Wait some time, Once it is timeout, Check whether the delayed RTP packet arrives.

If the delayed RTP packet does not arrive **then**

Indicate to the server that a RTP packet has been lost

else

Append this packetization data to the end of codestream of this image.

5.4 Immediate Feedback Mechanism

The conventional Real-Time Transport Protocol (RTP) provides periodic feedbacks on the quality of data distribution by using the RTP Control Protocol (RTCP). The RTCP is based on wired infrastructure networks, in which available bandwidth changes smoothly. Therefore, the periodic transmission of control packets is suffiecient to control the adaptive encodings and the speed of data distribution in traditional communications. However, the available bandwidth resource of wireless communications changes dramatically. The periodic feedback mechanism cannot accurately and promptly reflect the situation of wireless networks. To tackle this issue, we introduce our immediate feedback mechanism.

Unlike certain feedback mechanisms, which report periodically, the immediate feedback mechanism involves sending acknowledgments (ACK) for every received RTP packet. By keeping track of these acknowledgments, the server can establish a sense of the network situation. We will establish that this mechanism works well with wireless networks, in which the fluctuation in bandwidth is both drastic and unpredictable, since it gives the server the ability to react instantaneously to their variation. The ACKs are kept as small as possible with only their type, sequence number for ordering, and time stamp are assigned. The time stamp is used to approximate the round trip time (RTT) of each RTP packet. Missing acknowledgments are interpreted as dropped RTP packets.

The data required by the rate control algorithm to determine network status is provided by the immediate feedback mechanism.

5.5 Determining the Network Status

From the immediate feedback mechanism, the rate control algorithm can determine the round trip time and loss rate for every set of packets. Lost packets are treated as indicators of congestion; however, not all lost packets are caused by congestion. The unreliable nature of wireless networks also contributes to this loss. Thus, we employ a simple version of the loss differentiation algorithm proposed in [1]. When an ACK is missing, before considering the corresponding packets lost, we check the RTT of the next and preceding packets. If the difference is below a certain threshold, then the lost packet is attributed to the unreliability of the wireless network, not congestion.

The RTT is also considered an indicator of congestion on the wireless network. In our system, transmission delay is the determining factor in the RTT, since the difference of processing delay among clients is not significant. Furthermore, we limit the number of

hops on the wired network so that the wired-transmission delay is insignificant in comparison to the wireless-transmission delay. Therefore, a change in the RTT will reflect a change in the wireless network status. Our limitations on the wired network can be loosened by adopting a slower sampling rate of the client's navigational path. If we issue one request for a number of scenes, we will allow for a more flexible streaming, and thus less restrictions on the RTT and consequently on the wired network.

Since our streaming system can be employed on a wired-wireless network, the average RTT varies from client to client depending on its distance from the server. It also varies with one client from packet to packet depending on the number of hops the packet took on the wired network before reaching its destination. Thus, the RTT cannot be compared to an average RTT. However, for a set of RTP packets, their RTTs can indicate congestion if they increase steadily.

Based on the above, the network status is considered *congested* if lost packets are detected (subject to the loss differentiation algorithm) or RTTs are increasing for the set of examined packets. If no packets are lost and the RTTs are decreasing then the network is *unloaded.* The third scenario occurs when the RTTs remain relatively constant and no packets are lost. In this case, the network status is considered *constant* or stable.

The immediate feedback mechanism performs well in determining the network status for wired-cum-wireless networks. Like most network tracking mechanisms, immediate feedback utilizes both packet loss and RTT as indicators of network status. Subjecting packet loss to the differentiation algorithm eliminates confusing loss attributed to congestion to the unreliable nature of wireless networks. Employing RTT in our immediate feedback mechanism helps avoid packet loss by detecting packet buildup in

queues, which leads to a longer waiting time. Our calculations regarding RTT avoid comparing it with an experimental value since we recognize that clients have varying RTTs depending on how far they are from the server. Even with one client, two packets can take different routes and thus resulting in different RTTs. Recognizing this fact, we only attribute a difference in RTTs to fluctuating wireless bandwidth when it exceeds a certain threshold that takes into account the variations related to the different routes a packet might take on the wired network.

5.6 The Rate Control Algorithm

The rate control algorithm adjusts the rate based on the network status. If the status is congested, the rate is decreased. If it is unloaded, it is increased, and if it is constant, the rate is left unchanged. The amount by which the rate is incremented or decremented is crucial to our algorithm. It is pointless to increment the rate by an amount corresponding to a couple of extra packets. In other words, the system does not benefit if the increase in rate is analogous to a half, a quarter, or any other fraction of an image. It only makes sense to increment the rate if we can stream an extra image or two. Upon receiving a request, the server will determine the network status from the set of ACK of the previous request. If the network is unloaded, the rate is increased by an amount corresponding to one image. If the network is congested, the rate is decreased by an amount corresponding to one image.

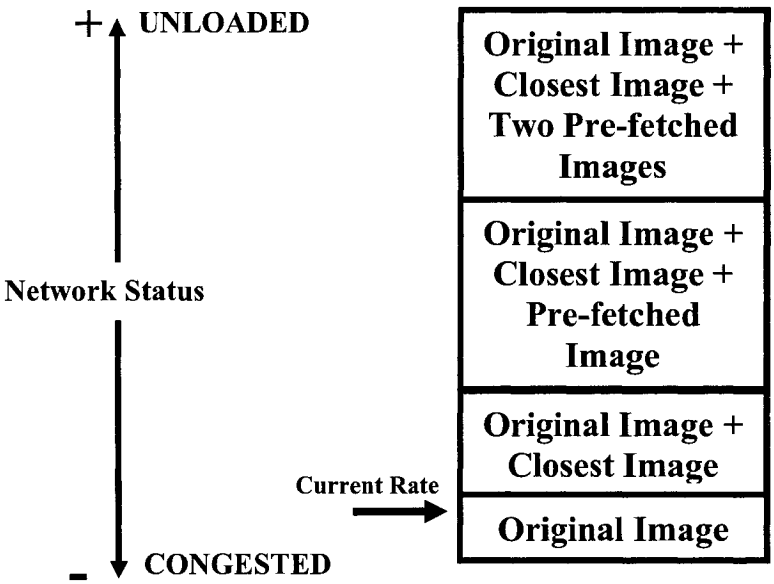

Figure 5.6: Rate Control Algorithm

The basic rate is enough to stream one image (requested by the client) within a time period T. This time period is determined by the client's rendering speed and desired frame per second (fps). For our system, rendering speed is limited to 100ms per image; thus, our T = 100ms (≈ 10 fps). The next level above the basic rate to which the control rate can increment corresponds to the streaming of two images (the requested image and the closest one to it). The closest image is streamed since it allows the rendering client to approximate the original image if the latter is lost or suffers from errors. Levels three and four correspond to streaming the original and closest images in addition to one pre-fetched image for level three and two pre-fetched images for level four.

The above model is also expandable; upper layers, such as three, four, or five pre-fetched images can easily be added on. Lower layers below the basic rate can also be added to the model. In such layers, only parts of the image are streamed, depending on

the priority packets set by the compression scheme. Thus, lower layers provide the original image with a quality less than that of the regular.

The rate control algorithm adjusts to the conditions of the network and maximizes the use of bandwidth to the best benefit of the streaming process. Using an entire image as a step increase or decrease in rate is significant enough to relieve a congested network or significantly benefit a requesting client. When increasing rate, any step corresponding to anything but a whole image or a multiple of it is meaningless. Transmitting an image and a fraction is the same as transmitting one image since that fraction will be discarded. On the other hand, decreasing the rate by a step corresponding to an image is a significant and aggressive reaction to bandwidth drop. When considering the case of the rate control mechanism overestimating the *unloaded* network status and streaming beyond the capabilities of the bandwidth, the packets expected to be dropped are those belonging to pre-fetched images. Thus, no effect will be detected on the client's side.

Although a one-view-per-request configuration was followed in this case, the server has the ability to allow the streaming of multiple views per client request. We recognize that the interaction between the client and server limits the streaming process. A server must wait for a request from the client before streaming the corresponding view. This one-view-per-request process will not scale when dealing with a high navigational speed or when aiming at higher frames per second. Depending on the application, the rate by which we sample a client's virtual position must correspond to more than one view in the world coordinate system. Thus, the client will issue a request for new views according to the sampling rate. We can modify our server to stream multiple views depending on the client's application and the rate by which it samples the virtual navigation.

5.7 Formal Implementation of Immediate Feedback and Rate Control

1) Immediate feedback mechanism

A. Implementation on Server:

Upon receiving an ACK:

Step A0. Parse RTP packet and extract sequence number and timestamp

Step A1. Add sequence number to list of successfully transmitted packets

Step A2. Calculate RTT and add to list of RTTs for recently transmitted packets

Step A3. **For** every recently transmitted packet **loop**

If sequence number of recently transmitted packet (P_i) doesn't exit in list of successfully transmitted packets **then**

If RTT of P_{i-1} - RTT of P_{i+1} > acceptable variation value **then**

Packets lost ← Packets lost + 1

Step A4. **If** Packet lost > 0 **then**

Network Status ← *Congested*

Else If RTT of last received packet - RTT of first received packet > Threshold **then**

Network Status ← *Congested*

Else If RTT of first last received packet - RTT of last received packet > Threshold **then**

Network Status ← *Unloaded*

Else

Network Status ← *Constant*

B. Implementation on Client:

Upon receiving an RTP packet:

Step B0. Parse RTP packet and extract sequence number and timestamp

Step B1. Set sequence number and time stamp for ACK

Step B2. Pass ACK to network layer

2) Rate Control Algorithm

Assumption: All images have sizes relatively close such that they can be divided into an equal amount of packets

Done once initially upon connection:

Step C0. Current Rate ← number of packets in an image

Upon receipt of request from client:

Step C1. **If** Network status → *Unloaded* **then**

If Current Rate < (4 * number of packets in an image) **then**

Current Rate ← Current Rate + number of packets in an image

ELSE If Network status → *Congested* **then**

If Current Rate > number of packets in an image **then**

Current Rate ← Current Rate - number of packets in an image

Step C2. Set interval time between two successive packet transmissions ← (T / Current Rate)

Chapter 6

Experiments and Results

In order to implement and test our streaming system, we required a framework for discrete event simulations. For this purpose, we chose to integrate ns2 (Network Simulator) in our system. The simulator provided the mechanism to simulate a wireless network environment. Our protocols were integrated into ns2 in order to be tested and to build the complete system.

The wireless network simulated for our experiments was a cellular network connected to a wired network via fifteen base stations covering an area of approximately 1500m x 1500m. The server lies on the wired network and is connected to the base station with high capacity links. The number of hops on the wired network does not exceed a certain threshold so that the associated delay does not affect the streaming experience. 801.11g was used for the Mac layer, and a total of fifteen clients roamed the network with an average speed of 10m/s. The clients were restricted to one cell at one point to test the performance of the rate control mechanism under shared bandwidth.

The navigation path was written in OTcl script and represents the navigation of a client within a room with a long rectangular table situated in the middle. The client walks into the room, around the table, and toward the end of the table in order to find an empty seat. It generates a request for images from the client based on this pre-determined path. The navigational path contains a reasonable amount of change in navigation mode, with image requests being sent at a rate of approximately ten per second. We tested the performance of the system with one to fifteen clients simultaneously carrying out

streaming sessions with the server. The streaming session for each client was continuous, with up to 300 images being streamed.

6.1 Performance metrics

In order to evaluate the performance of our proposed algorithms, we made use of some important performance metrics to test the quality and speed of our streaming as well as the performance of our pre-fetching mechanism. We chose the following performance metrics: *drop rate, burst length, duration, average delay time, average number of images in cache, average client cache hits, average pre-fetched images in cache,* and *ratio of cache hits to pre-fetched images in cache.*

The quality of streaming was determined by the *drop rate* as well as the *burst length* and *duration*. The *drop rate (DR)* indicates the proportion of lost packets to total packets. This metric is related to three factors: network condition, rate control, and packetization scheme. The better the network condition, the lower the *DR*. By constantly keeping track of network status and adjusting the streaming rate accordingly, we can limit the amount of discarded packets. In addition, the packetization scheme can affect this performance metric. For example, if the packetization item is parsed in a good manner and an appropriate length of a packet is applied, the *drop rate* will decrease.

The *burst length and burst duration* represent the burst performance of the real-time transport protocol. A burst is a period during which a high proportion of packets are lost. The *burst length* stands for the number of lost packets in the burst period. The *burst duration* indicates how long the burst period lasts.

The speed of our streaming is determined by the *average delay time* that the client must to wait before receiving the requested image, as well as the *average request time*

that the client waits to receive both the requested and pre-fetched images, if any. The smaller these metrics are, the better the streaming and the higher the frame per second rate.

To test the performance of the pre-fetching mechanism, we also kept track of the *average number of images* in a client's cache since this is a good indicator of how well the rate control and pre-fetching mechanism work together to maximize bandwidth utilization. The *average client cache hits* give us a good sense of how the client benefits from its caching mechanism. A hit is defined as the time when a client finds the requested image in the cache, and thus does not request it from the server. The *average pre-fetched images in cache* is a good indicator of how the pre-fetching mechanism is affected by limiting resources. Finally, we can keep track of how useful the pre-fetched images are to the client by examining the *ratio of cache hits to pre-fetched images in cache.*

6.2 Experimental Results and Analysis

Examining the results from Figure 6.1 for the *average number of images in the cache*, we can realize the adaptability of the rate control mechanism in reacting to the network status. When one client was connected to the server, the amount of images ending up in the client's cache rose to 309 for only 150 requests. The control mechanism was aware of the available bandwidth and was quick to utilize it by sending additional pre-fetched images. If the pre-fetching was not implemented, the number of images at the client side would not exceed 150 (ratio of 1 image per request). Having such a large number of images for a small number of requests helps make streaming run more smoothly. When considering fifteen clients connected to the server, available bandwidth drops, along with the ability of the server to stream more pre-fetched images. Again, the

rate control mechanism was able to adjust to the new scenario and lower the streaming rate. For the same number of requests (150), the server only managed to stream 176 images; this is almost half the number of those streamed when one client was connected to the server.

Since the average number of images in the client's cache decreases with the increase of the number of clients, it is obvious that the chances of a client finding the requested image in its cache decrease. The *average client cache hits* from figure 6.8 decrease as the bandwidth becomes more limited. The same reasoning can be applied to the *average pre-fetched images in cache*, with their numbers decreasing as the number of clients increases (Figure 6.9).

Examining the *ratio of cache hits to pre-fetched images in cache* in Figure 6.10, we notice that this ratio maintains a steady value with the decrease of network resources. This means that regardless of networks status, whenever images are pre-fetched, the chances of them being used at least once are very high.

The drop rate is one of the most important performance metrics when dealing with wireless communications. For our system, the drop rate varied from 0.67% to 4.52% when one to fifteen clients were connected (Figure 6.2). Even when considering the worst case, the drop rate remained at a very reasonable value. The control mechanism's periodic adjustments per request were the reason for the low rate of lost packets. The scheme was also able to detect a drop in bandwidth before it reached the critical point at which packets are dropped.

The *average delay time* varies between 20 and 60 ms (Figure 6.3) but remained well below the 100ms threshold required by our system to render an image. By

maintaining this small delay, we can guarantee a good frame per second, which can reach more than 10 fps at some points. Examining *average request time* (Figure 6.4), we notice that as the number of clients increases, the request time increases significantly. However, we must consider that these requests do not merely carry with them the original image, but sometimes pre-fetched ones as well. Thus, if the prediction proves to be correct, the next request will not be sent by the client since the image will have already been streamed. This means that the 100 ms threshold does not necessarily apply to this metric since the streamed images may suffice for more than one request depending on the accuracy of the path prediction and pre-fetching mechanism.

Considering the *burst length*, in our experiments the value remained stable at around 3-4 packets. Thus, in the worst-case scenario, the burst will constitute 1-2 damaged images, but at the operating streaming rate the client should not notice the difference. In Figure 6.5, the *burst duration* reaches 120ms in the worst case; this is still an acceptable value for our system.

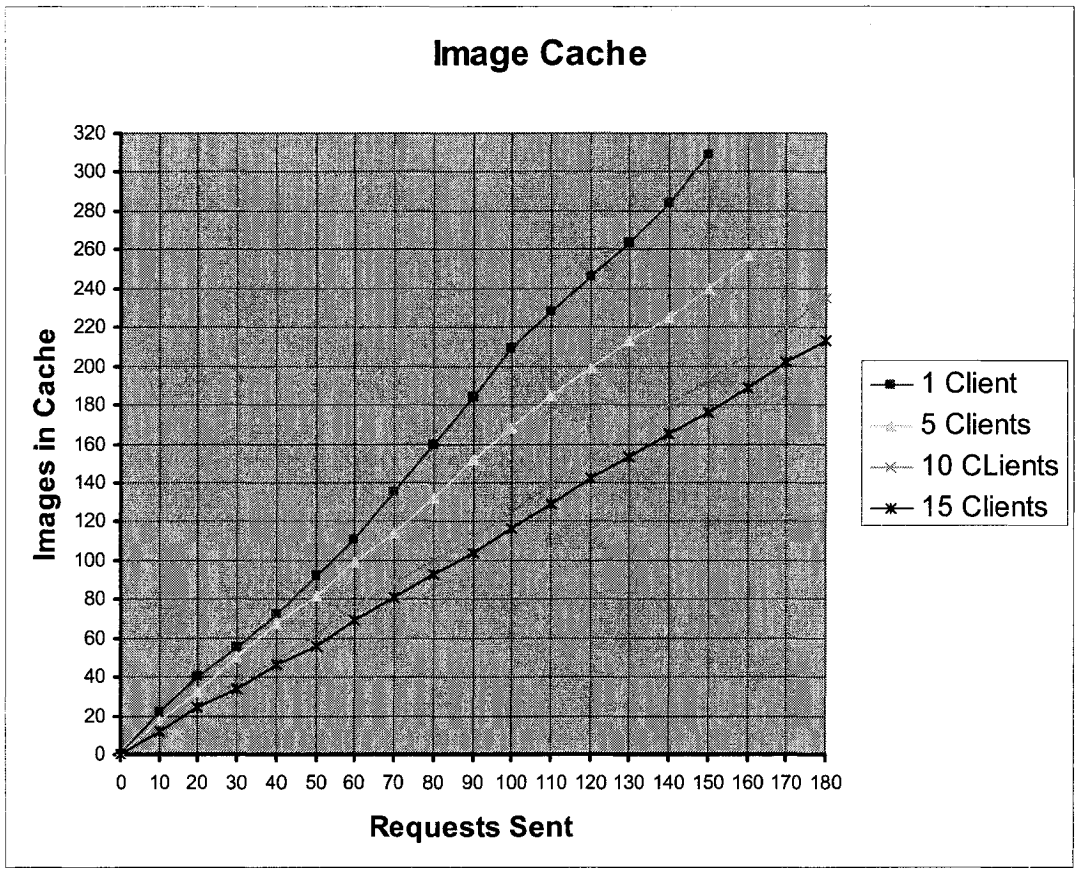

Figure 6.1: Average Number of Image in Cache

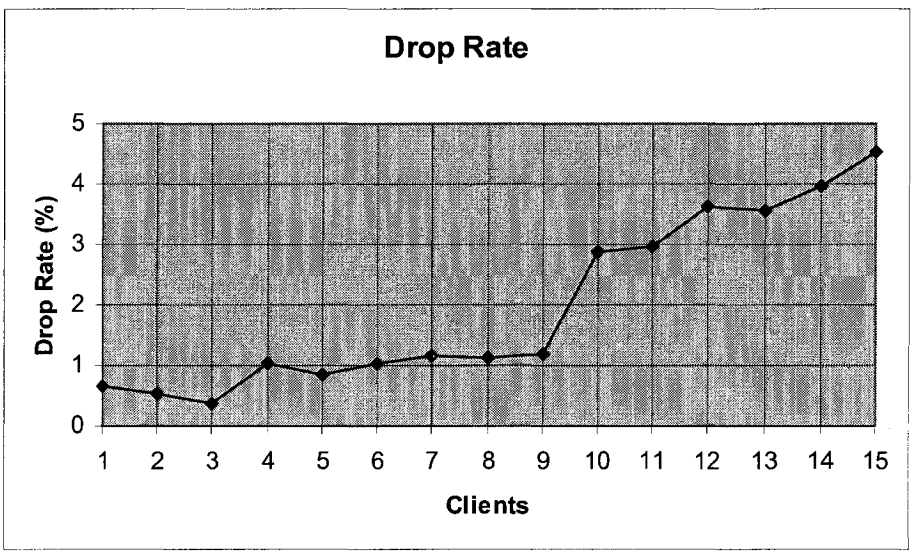

Figure 6.2: Average Drop Rate

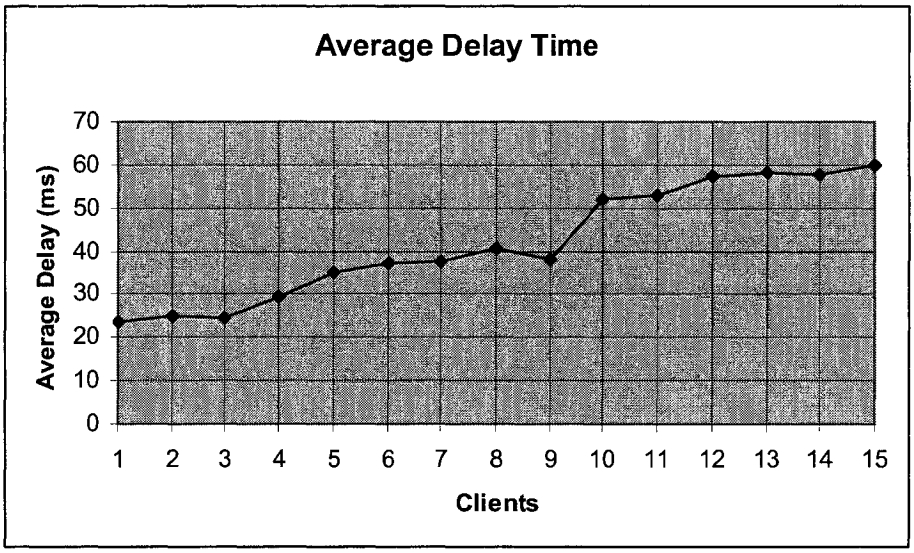

Figure 6.3: Average Delay Time

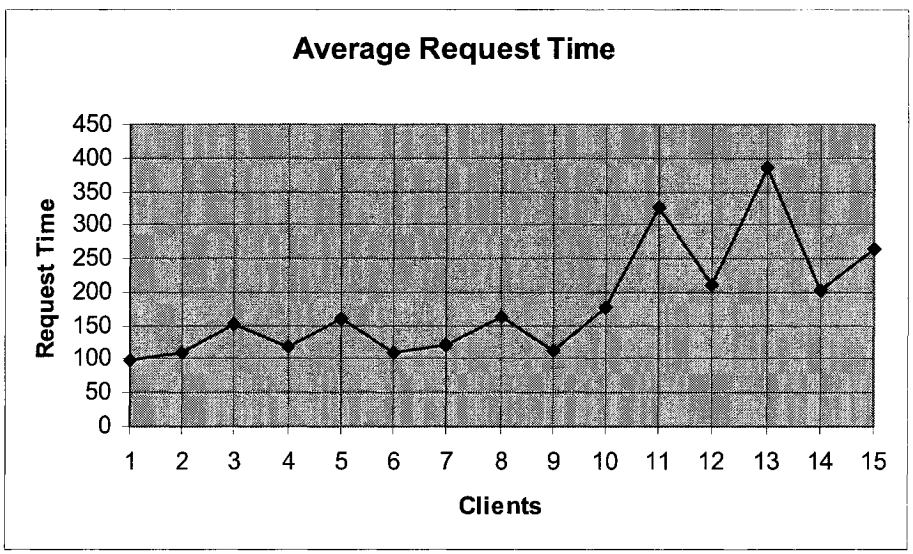

Figure 6.4: Average Request Time

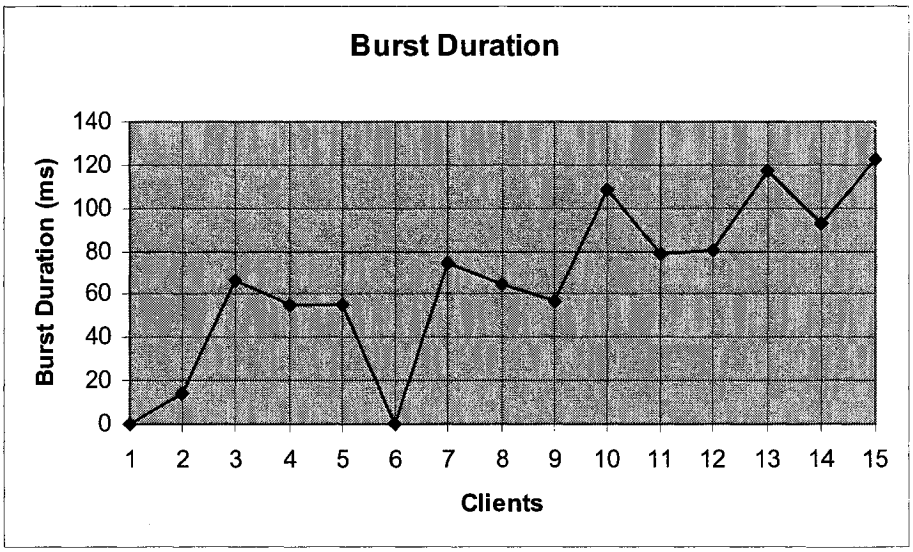

Figure 6.5: Average Burst Duration

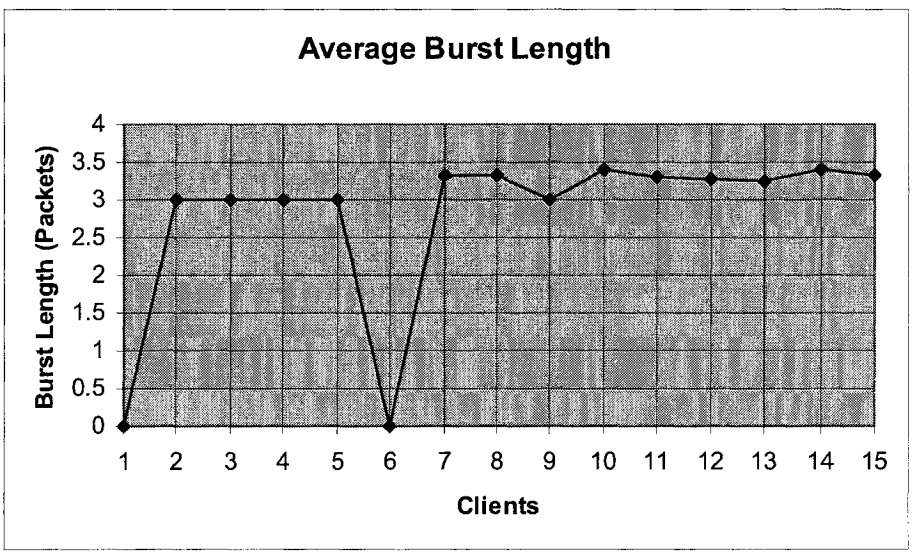

Figure 6.6: Average Burst Length

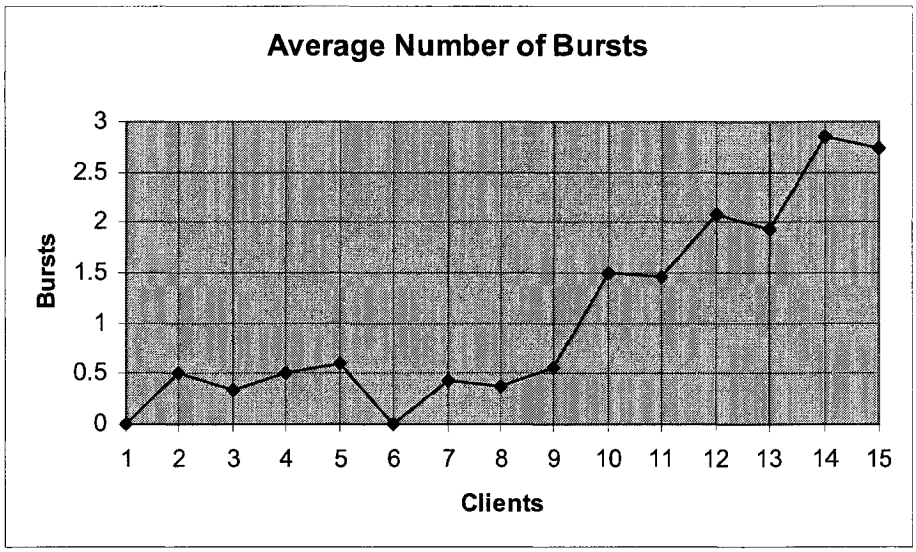

Figure 6.7: Average Number of Bursts

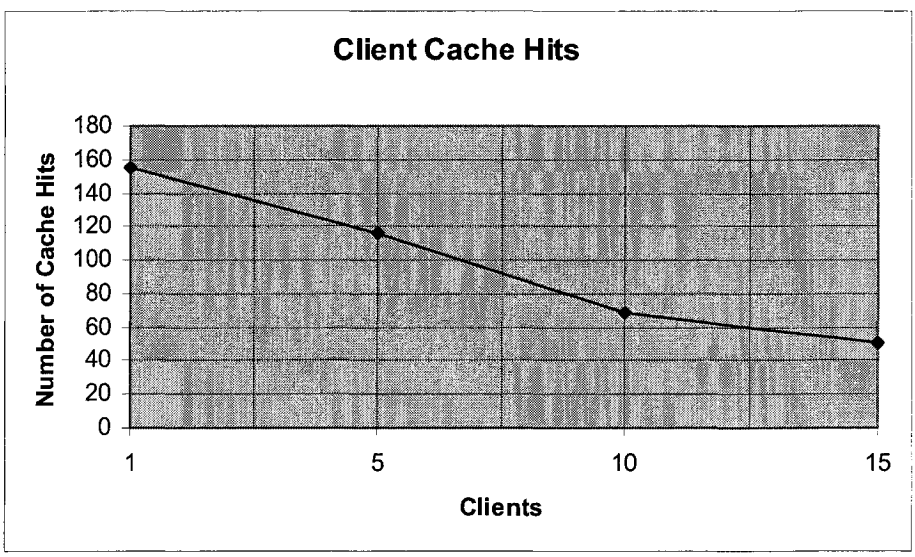

Figure 6.8: Average Cache Client Hits

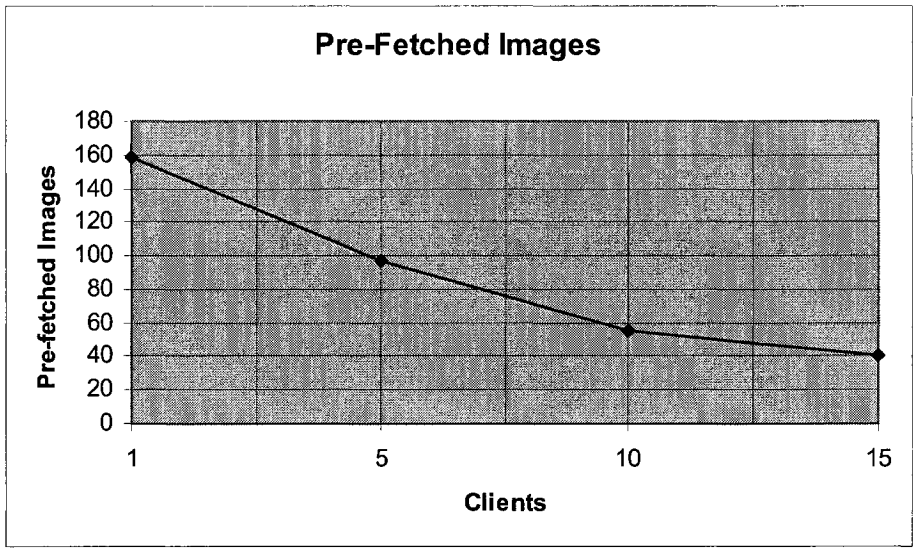

Figure 6.9: Average Pre-fetched Images in Client Cache

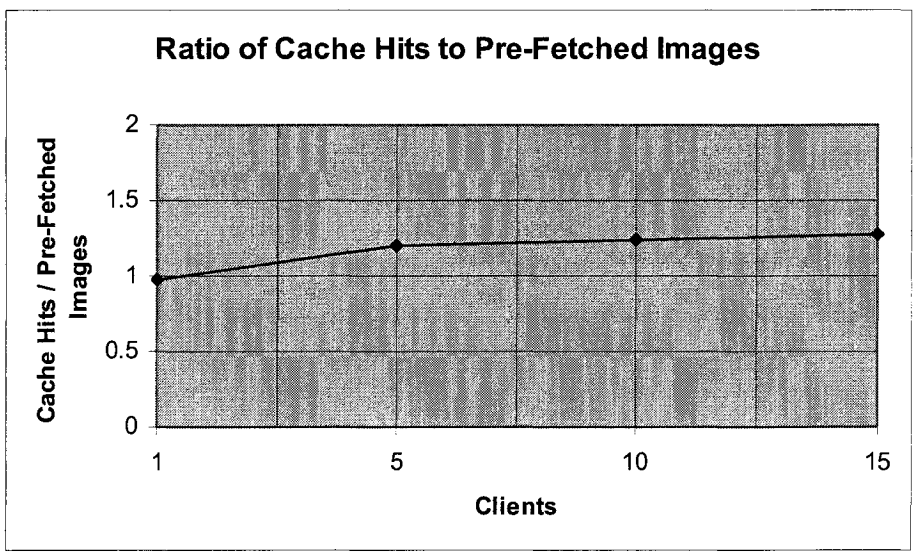

Figure 6.10: Ratio of Client Cache Hits to Pre-fetched Images

Chapter 7

Conclusion

Image-based virtual environments are the subject of many research endeavors; however, it is extremely challenging to stream 3D scenes over wireless communications because of the huge volume of data and error-prone wireless communications. Interactivity also adds restrictions to streaming, making it an on-demand process based on the client's navigational path in the virtual environment.

In the following section we will highlight our main contributions and identify several directions for future work.

7.1 Summary of Contributions

The contributions of this thesis are as follows:

5. *System architecture for interactive streaming of image-based virtual environments* is presented based on the client server paradigm. We simplified certain aspects of 3D scenes over wired and wireless networks because of the tremendous volume of data and complex computation. We used the simplest representation of 3D scenes, a 2D image collection, and we adopted parallel image morphing to render any novel image on the client side. A specific compression algorithm called Morphing- JPEG was utilized, as well as multiple functional layers to handle multiple streaming sessions to clients.

6. *An interactive streaming protocol*: A signaling protocol is developed for the purpose of exchanging interactive requests and responses between streaming servers and clients through controlling multiple streaming sessions of image-based 3D scenes.

7. *A real-time transport protocol (RTP)*: A unique real-time transport protocol payload format is required to be developed for a specific multimedia compression and multimedia application scenario. We proposed a real-time transport protocol for image morphing. We designed a series of necessary fields for RTP header structure and presented a packetization scheme.

8. *An adaptive rate control algorithm, an immediate feedback mechanism, and a path prediction and pre-fetching scheme*: An adaptive rate control algorithm which uses an immediate feedback mechanism to determine the network status was proposed. The network status is constantly monitored to allow for the quick reaction of the system to the network nature. Both packet loss and round-trip time are used as indicators of whether or not a network is congested. We also introduced our path prediction and pre-fetching mechanism, which aims, along with rate control, to fully utilize the available bandwidth of the wireless network.

7.2 Future Work

We can identify several directions for further research:

- We shall perform an analytical study of the types of applications that might utilize our streaming system and further discuss its compatibility with their specific requirements. We also plan to examine how the path prediction and pre-fetching scheme can benefit from application dependent information.

- We plan to extend our streaming system to support ad-hoc networks and propose a real time transport protocol that can handle its specific requirements.

References

[1] C-M. Huang, Y-T. Yu, and Y-W Lin, "An Adaptive Control Scheme for Real-time Media Streaming over Wireless Networks," 17th International Conference on Advanced Information Networking and Applications, Page(s):373 – 378, March 2003

[2] H. Wu, Q. Zhang, and W. Zhu, "Design Study for Multimedia Transport Protocol in Heterogeneous Networks," IEEE International Conference Communications, Page(s):567-571, vol.1. May 2003

[3] K. Tan, Q. Zhang, W. Zhu, "An End-to-end Rate Control Protocol for Multimedia Streaming in Wired-cum-wireless Environments," Proceedings of the 2003 International Symposium on Circuits and Systems, Page(s):II-836 - II-839 vol.2 May.

[4] O.B. Akan, I.F. Akyildiz, " ARC: The Analytical Rate Control Scheme for Real-time Traffic in Wireless Networks," IEEE/ACM Transactions on Networking, Volume 12, Issue 4, Page(s):634 – 644, Aug. 2004

[5] V. Zagorodnov, P.J. Ramadge, "Data Rate Smoothing in Interactive Walkthrough Applications using 2D Prefetching," International Conference on Image Processing, Volume 3, Page(s):III-201 - III-204 vol.3, June 2002

[6] C. Grunheit, A. Smolic, T. Wiegand, "Efficient Representation and Interactive Streaming on High-Resolution Panoramic Views," International Conference on Image Processing. Page(s):III-209 - III-212 vol.3, June 2002

[7] http://amp.ece.cmu.edu/projects/mobilecamarray/.

[8] http://graphics.stanford.edu/projects/array/.

[9] http://h10010.www1.hp.com/wwpc/us/en/sm/wf05a/215348-64929-215381-314903-f66-322916.html.

[10] http://www.isi.edu/nsnam/ns/.

[11] http://www.japan-mobile-net.com/themes/phone.htm.

[12] http://www.virtuallyvancouver.com/index2.html.

[13] E. H. Adelson and J. R. Bergen, "The Plenoptic Function and the Elements of Early Vision". Computational Models of Visual Processing, Chapter 1, Edited by Michael Landy and J. Anthony Movshon. The MIT Press, Cambridge, Mass., 1991.

[14] Tomas Akenine-Moller and Eric Haines, "Real-time Rendering" second edition. A K Peters Natick, Massachusetts, 2002.

[15] Daniel G. Aliaga and Ingrid Carlbom, "Plenoptic Stitching: A Scalable Method for Reconstructing 3D Interactive Walkthroughs". Proceedings of the 28th annual conference on Computer graphics and interactive techniques, August 2001.

[16] L. Berc, W. Fenner, R. Frederick, and S. McCanne, RFC2035: RTP Payload Format for JPEG-compressed Video. Standards Track, Network Working Group, October 1996.

[17] George Chen, Li Hong, Kim Ng, Peter McGuinness, Christian Hofsetz, Yang Liu, and Nelson Max, "Real-time Rendering: Light Field Duality: Concept and Applications", In Proceedings of the ACM symposium on Virtual reality software and technology, November 2002.

[18] Shenchang Eric Chen and Lance Williams, "View interpolation for image synthesis" In Proceedings of the 20th annual conference on Computer graphics and interactive techniques, September 1993.

[19] Eric Edwards, Satoshi Futemma, Nobuyoshi Tomita, and Eisaburo Itakura, "RTP Payload Format for JPEG 2000 Video Streams", Internet Engineering Task Force (IETF), "draft-ietf-avt-rtp-jpeg2000-05.txt", July 2004.

[20] Yi-Ping Hung Tse Cheng Ho-Chao Huang, Shung-Hua Nain. "Disparity-based View Morphing: A New Technique for Image-based Rendering". In Proceedings of the ACM symposium on Virtual reality software and technology, November 1998.

[21] D. Hoffman, G. Fernando, V. Goyal, and M. Civanlar. "RFC2250: RTP Payload Format for MPEG1/MPEG2Video". Standards Track, Network Working Group, January 1998.

[22] ISO/IEC. "Coding of Moving Pictures and Associated Audio for Digital Storage Media up to about 1.5mbits/s". ISO/IEC International Standard 11172, November 1993.

[23] Elysium Ltd and 2KAN. http://www.jpeg.org/jpeg2000/.

[24] Leonard McMillan and Gary Bishop, "Plenoptic Modeling: An Image-based Rendering System, Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, September 1995.

[25] William B. Pennebaker and Joan L. Mitchell, "JPEG: Still Image Data Compression Standard", Van Nostrand Reinhold, February 1993.

[26] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-time Applications", Standards Track, Network Working Group, January 1996.

[27] H. Schulzrinne and S. Petrack, "RFC2833: RTP Payload for DTFM Digits, Telephony Tones and Telephony Signals". Standards Track, Network Working Group, May 2000.

[28] H. Schulzrinne, A. Rao, R. Lanphier, M. Westerlund, and A. Narasimhan, "Real-time Streaming Protocol (RTSP)", Internet Draft, Internet Engineering Task Force, February 2004.

[29] S. M. Seitz and C. R. Dyer, "View Morphing", In Proc. SIGGRAPH 96, pages 21–30, 1996.

[30] Heung-Yeung Shum and Li-Wei He, "Rendering with Concentric Mosaics", In Proceedings of the 26th annual conference on Computer graphics and interactive techniques, July 1999.

[31] KSVO the Silicon Valley Sentinel-Observer's NetTV Channel, "Panorama - A Better Way to See All Around", In ACM SIGCAS Computers and Society, 1998.

[32] T. Turletti and C. Huitema, "RFC2032: RTP Payload Format for H.261 Video Streams", Standards Track, Network Working Group, October 1996.

[33] International Telecommunication Union, "Video Codec for Audiovisual Services at P x 64 Kbit/s", ITU-T Recommendation H.261, 1993.

[34] International Telecommunication Union, "Video Coding for Low Bit rate Communication", ITU-T Recommendation H.263, 1993.

[35] C. Zhu, "RFC2190: RTP Payload Format for H.263 Video Streams", Standards Track, Network Working Group, September 1997.

[36] R. Zopf, "RFC3389: Real-time Transport Protocol (RTP) Payload for Comfort Noise". Standards Track, Network Working Group, September 2002.

[37] Stephan Brumme, "Multimedia Streaming On Mobile Phones", Lecture on Advanced Data Communications at the University of Technology, Sydney, http://www.stephan-brumme.com, 2004

[38] John G. Apostolopoulos, Wai- tian Tan, Susie J. Wee, "Video Streaming: Concepts, Algorithms, and Systems", Mobile and Media Systems Laboratory HP, http://www.hwswworld.com/downloads/, 2002.

[39] Edouard Lamboray, Stephan Würmlin, Markus Gross, "Real-Time Streaming of Point-Based 3D Video", IEEE Proceedings on Virtual Reality, March 2004.

[40] Sheng Yang, C.-C. Jay Kuo, "Robust Graphics Streaming in Walkthrough Virtual Environments via Wireless Channels", IEEE GLOBECOM 2003.

[41] Fredrik Montelius,Oscar Larsson, "Streaming Video in Wireless Networks, Service and Technique", Master Thesis, Linköping Department of Electrical Engineering, Sweden. JVC Streaming Solution, White Paper, March 2004.

[42] Stephan Olbrich, Helmut Pralle, "A Tele-immersive, Virtual Laboratory Approach based on Real-Time Streaming of 3D Scene Sequences", Proceedings of the ninth ACM international conference on Multimedia, 2001.

[43] Toshiki Hijiri, Kazuhiro Nishitani, Tim Cornish, Toshiya Naka and Shigeo Asahara, "A Spatial Hierarchical Compression Method for 3D Streaming Animation", Proceedings of the fifth symposium on Virtual reality modeling language, ACM, 2000

[44] Krishna Sajja, Dr. Pamela Lawhead, "Providing Streaming Capabilities to Virtual World Applications", The University of Mississippi, USA, http://geoworkforce.olemiss.edu/student%20papers/Krishna_Paper.pdf.

[45] S. Olbrich, H. Pralle, "Virtual Reality Movies – Real-Time Streaming of 3D Objects", TERENA-NORDUnet Networking Conference (TNNC), 1999

[46] Peter Eisert, "Immersive 3-D Video Conferencing: Challenges, Concepts, and Implementations" Proc. SPIE Visual Communications and Image Processing (VCIP), Lugano, Switzerland, July 2003.

[47] S. Olbrich1, H. Pralle, "Using Streaming and Parallelization Techniques for 3D Visualization in a High-Performance Computing and Networking Environment",

Proceedings of the 9th International Conference on High-Performance Computing and Networking, 2001.

[48] Ho-Chao Huang, Shung-Hua Nain, Yi-Ping Hung and Tse Cheng, "Disparity-Based View Morphing- A New Technique for Image-Based Rendering", http://citeseer.ist.psu.edu/85340.html, 1998

[49] Shenchang Eric Chen, Lance Williams, "View Interpolation for Image Synthesis", Proc. Conference on Computer Graphics, ACM, 1993

[50] Nicholas Appleby, Dr. Patrick Marais, "Scalable Model Viewing: A PDA Based Client Architecture for Remote Viewing of Complex 3D Scenes", Technical Report, http://pubs.cs.uct.ac.za/archive/00000061/, 2003

[51] Ismo Rakkolainen, Teija Vainio, "A 3D City Info for Mobile Users", http://citeseer.ist.psu.edu/rakkolainen00city.html, 2000

[52] Vladimir Stankovi´c, Raouf Hamzaoui, Zixiang Xiong, "Live video streaming over packet networks and wireless channels", http://www.inf.uni-konstanz.de/cgip/bib/files/StHaXi03.pdf, 2003

[53] Yu Lei; Xiaolin Zheng; Zhongding Jiang; Chen, D.; "Adaptive Streaming Panoramic Video for Image Based Rendering System", Networking, Sensing and Control, 2005.

[54] J.Li, H.H.Sun; "On Interactive Browsing of Large Images", IEEE Transactions on Multimedia, VOL.5, N0.4, pp.581-590, December 2003