TEEVE: The Next Generation Architecture for Tele-immersive Environments

Zhenyu Yang, Klara Nahrstedt, Yi Cui, Bin Yu, Jin Liang
University of Illinois at Urbana-Champaign
Department of Computer Science
SC, 201 N. Goodwin, Urbana, IL 61801
Email: {zyang2, klara}@uiuc.edu
Telephone: (217)333-1515

Sang-hack Jung, Ruzena Bajscy
University of California at Berkeley
Department of EECS
253 Cory Hall, Berkeley, CA 94720
Email: {sangj, bajcsy}@EECS.Berkeley.EDU

Abstract

Tele-immersive 3D multi-camera room environments are starting to emerge and with them new challenging research questions. One important question is how to organize the large amount of visual data, being captured, processed, transmitted and displayed, and their corresponding resources, over current COTS computing and networking infrastructures so that "everybody" would be able to install and use tele-immersive environments for conferencing and other activities. In this paper we propose a novel cross-layer control and streaming framework over general purpose delivery infrastructure, called TEEVE (Tele-immersive Environments for EVErybody). TEEVE aims for effective and adaptive coordination, synchronization, and soft QoS-enabled delivery of tele-immersive visual streams to remote room(s). The TEEVE experiments between two tele-immersive rooms residing in different institutions more than 2000 miles apart show that we can sustain communication of up to 12 3D video streams with 4∼5 3D frames per second for each stream, yielding 4∼5 tele-immersive video rate.

Keywords: Tele-immersion, System Design

I. Introduction

Tele-immersive 3D multi-camera room environments are starting to emerge and will allow for a more effective social interaction among distributed parties. To allow the usage of these tele-immersive environments for different audiences, these environments need to be able to run on general purpose infrastructure such as Windows or Linux, and Internet protocols that we have today. For a broader audience ("everybody"), it is not realistic to assume that the underlying computing and networking infrastructure will change rapidly. On the other hand, we observe that

the advances of end-devices (e.g., 3D cameras, plasma displays) that make the tele-immersive edge applications possible are becoming available and deployable.

Hence, one of the challenges is *"how do we bring these tele-immersive edges together with the general purpose infrastructure"* so that broader audience can enjoy these room environments for business and entertainment. This challenge is exacerbated by the amount of data that the tele-immersive 3D camera array edges produce. For example, in our environment, one 3D camera produces around 3Mbytes of uncompressed data per second.

There have been various interesting and important efforts and results to create tele-conferencing and tele-immersive environments (e.g., [1], [2], [3], [4], [5]). The current approaches represent a very good start for the next generation of tele-immersive systems where the ultimate goal is to give to the broader audience 3D tele-immersive experience over their available computing and communication infrastructure. However, the existing solutions either do not provide 3D multi-camera tele-immersive content (e.g., NetMeeting, Polycom) or require modifications to the available OS and network infrastructure for support of 3D video content which might be difficult to implement in case of broader audience (e.g., Coliseum, Virtual Amphitheater, cluster-to-cluster applications).

In this paper, we investigate the design space between the 3D multi-camera/multi-display tele-immersive edges and the general purpose computing and communication infrastructure available today. The design space includes application functions for capturing, reconstructing and displaying 3D video content, as well as the distributed middle-ware functions for compressing, coordinating, synchronizing and streaming the 3D content over Internet2. Our design space is based on the cross-layer approach and represents the foundation of our TEEVE (Tele-immersive Environments for EVErybody) framework. TEEVE provides integrated solutions to challenges such as (1) real-time 3D video reconstruction and compression, (2) Quality-of-Service (QoS)-aware coordination and synchronization of multiple 3D video streams, (3) multi-tier QoS-aware protocol for 3D video streaming, and (4) association between 3D content and multi-view displays. Our experiments are encouraging and indicate that TEEVE is bringing us close to true collaborative environments for "everybody".

The paper introduces the overall system architecture of TEEVE including the description of the model, the integration of various components and the preliminary tele-immersive streaming experiment. Our next step will focus on performing the user study. Section II describes related work in the areas of teleconferencing and tele-immersive environments. To present our TEEVE framework, we first discuss the multi-tier application model, consisting of 3D video data and timing models, in Section III. In Section IV, we present the TEEVE architecture taking the top down approach to introduce the user, layer and device views. Since the overall design of TEEVE solutions is based on the cross-layer approach between application and service middle-ware layers, we discuss TEEVE solutions across two major communication planes, the control plane and the data transport plane. In Section V, we

introduce functions, services and protocols to setup, coordinate and adapt TEEVE. Section VI presents the multi-tier end-to-end streaming protocol. In Section VII, we validate TEEVE performing extensive LAN/WAN experiments. Section VIII concludes by discussing our findings with the TEEVE system.

II. Related Work

We divide related work into three categories: (a) teleconferencing systems over COTS computing platforms and existing Internet. (b) teleconferencing systems relying on augmented OS/middleware support and, (c) teleimmersion systems over advanced networking service such as Internet2.

Most video conferencing systems widely in use today fall into the first category. Examples include PolyCom, Microsoft NetMeeting, WebEx, etc. These systems aim to provide point-point or limited multi-point communication for desktop users. Only a single view is available for each user through a 2D web camera. Since such systems produce medium or low quality video and audio stream compressed using standardized media format (H.263, MPEG4, etc.), the bandwidth requirement is largely compatible with a variety of networking environments including DSL and ISDN connections.

Solutions in the second category aim to provide users the telepresence experience beyond the single 2D view. In [2] and [3], participants of video conference are grouped and tiled into synthetic environments such as virtual auditorium or digital amphitheater. Coliseum [1] is a desktop-based immersive conferencing system. Here, the 3D view of a user is captured by multiple cameras, extracted from background, then reconstructed and embedded into the virtual environment. Finally, 2D video is created by locally rendering the 3D scene from the user-defined viewpoint, then streamed to the network. A commonality shared by these applications is that the transport services of current commodity operating systems (e.g., Windows) do not satisfy their demands for the media streaming purpose. Therefore, advanced software modules or middleware framework are developed to enhance the existing transport service to effectively support these applications. In [2] and [3], frame tiling and stream merging functions are provided to produce synthetic media stream. [1] created Nizza, a middleware framework to simplify the development of the media streaming subsystem.

Solutions in the last category include Virtue [6], MetaVerse [7][5], and the National Tele-Immersion Initiative [8][4][9]. These systems aim to provide teleimmersive realism to users. Similar to Coliseum [1], they rely on multiple cameras to capture the 3D scene. However, after reconstruction, the 3D data stream, instead of the 2D rendered view, is transmitted to the network. The advantage of this choice is to give users maximum freedom to watch the 3D scene from any viewpoint and change it anytime he/she wishes to do so. In case more than one user are present at a local site such as a conferencing room with multiple displays, they can also share the same 3D video stream by watching it from different viewpoints. Due to the huge volume of 3D data stream, it

poses significant challenges to the current transport service and the capacity of the traditional Internet infrastructure itself. The Tele-Immersion system developed by UNC Chapel Hill is deployed over the Internet2, in order to cope with its considerable traffic demand. The entire architecture of the network transport service also needs complete innovation. [10] proposed a cluster-to-cluster architecture. Here, gateways are inserted at both ends of the path, which aggregate and regulate all data flows through them. This solution was shown to well address the synchronous arrival and racing condition among multiple camera streams. In [5], an overlay-based multi-path routing service was introduced for efficient delivery of multiple streams. It is shown that by utilizing redundant paths provided by the overlay routing service, the aggregate bandwidth is significantly augmented, in comparison to the case of single end-to-end path. Other than attempts to modify the network transport service, many efforts are also made to design novel 3D data compression techniques. In [8], a 3D data compression scheme was proposed, which exploits the fact that many pixels in the 3D space are captured by multiple cameras. In order to remove these redundant points, one reference stream is chosen, which is used by other streams to remove the redundant pixels already appearing in the reference stream.

Compared with previous research efforts, the contribution of this paper is the design, implementation and validation of a flexible and end-to-end cross-layer architecture to incorporate advances of tele-immersive techniques and devices aimed for high quality immersive experience while relying on general purpose computing and networking infrastructures.

III. TEEVE Application Model

The TEEVE application is modeled as a distributed multi-tier application (Figure 1). The first tier is the *capturing tier* that consists of a set of 3D camera clusters organized in a half-circle within a room, each camera cluster reconstructing one 3D video stream. The 3D video streams are then forwarded over LAN to service nodes for video compression. The second tier is the *transmission tier* that takes the compressed video streams, shapes them and sends them over Internet2 to the receiving service nodes within a remote room. The third tier is the *displaying tier* which takes the received 3D video streams, decompresses them, renders them into an immersive video and forwards the rendered video over LAN to one or more displays. Figure 2 presents a pictorial view of a possible tele-immersive environment with 3D camera clusters and displays.[1] Each tier of the TEEVE framework deploys algorithm, service and protocol that require clear and sound application data and timing models.

[1]The picture presents an early version of our TEEVE testbed; at the current point we have expanded the number of 3D camera clusters towards 6 3D cameras in one location and 12 3D cameras in remote location.

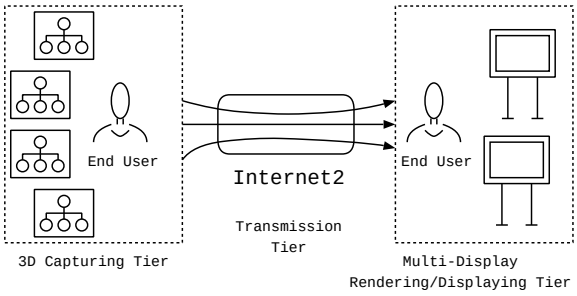


Fig. 1. TEEVE Application Model

Fig. 2. TEEVE Environment

A. TEEVE Data Model

The overall TEEVE data model consists of two parts: (a) the 3D reconstructed video data model that represents the information coming out from a single 3D camera cluster, and (b) the integrated data model that includes all 3D video images that capture the scene from different 3D camera clusters at the same time.

1) 3D Video Data Model: The data from one camera cluster is the 3D frame information (f) of a scene which can be obtained by applying stereo algorithms on images captured from multiple viewpoints. In general, given two or more images of a scene, depth of each point in 3D space can be obtained by finding matching points on different images and using triangulation.

The 3D information is created by a set of four cameras that form a basic processing unit which is used to capture both color and depth information from each viewpoint. The synchronization of the cameras is achieved by connecting them via hotwires. More specifically, three black and white cameras are used to compute depths by adopting a trinocular stereo algorithm ([11], [12]) and a color camera is used to extract appearance information from the corresponding viewpoint. A single camera cluster computes depth and color of each point on the reference image which is captured by a central black and white camera. The data that is returned by each unit cluster consists of an array of $XYZRGB$ (i.e. depth coordinates and color) information for each pixel. Each pixel requires 5 bytes to store RGB (3 Bytes) and XYZ (2 Bytes) information. Thus, assuming that w and h denote width and height of an image in pixel, the size of raw data can be calculated by: $size(f) = w \times h \times resolution/pixel$. In our environment, each single camera cluster can reconstruct a 3D frame f at the rate of 10 fps, and if we consider $w = 320$ pixels, $h = 240$ pixels, then one 3D video stream, uncompressed, demands 3,840,000 Bytes per second.

2) Integrated Data Model: The capturing tier consists of a set of equal N 3D camera clusters organized in a half-circle (or circle)[2] and each camera cluster captures one 3D video image at a time T_{cap}. This means that at time T_{cap} the capturing tier must have N 3D reconstructed frames that constitute a *macro-frame* to show the same scene from different view points of the room. This macro-frame F consists of $(f^1, ..., f^N)$ single 3D frames from the individual camera clusters. Hence, the tele-immersive application yields a stream of macro-frames $(F_{begin}, ..., F_{end})$ that are captured at the *macro-frame-rate* 10 Fps, and *macro-frame-period* of 100 ms in our environment.

B. Timing Model

It is important to realize that although one macro-frame consists of N 3D reconstructed frames (f^i) at time T_{cap}, they cannot be transmitted at the same time since their simultaneous transmission would cause a strong congestion, especially in the second tier of the TEEVE application. Hence, the individual frames f^i of the macro-frame F need to be shaped and spaced out within a certain completion time interval $T_{send(j)}$ ($T_{send(j)}$ should be less than the macro-frame-period). Furthermore, it is important for rendering of the overall macro-frame at the displaying site that the individual frames f^i arrive within a completion time interval $T_{recv(j)}$ which is either equal to $T_{send(j)}$ or $(T_{recv(j)} - T_{send(j)}) \leq \delta$, where δ is small. At the displaying site, all frames f^i $(i = 1, ..., N)$ will be then received and rendered together into the resulting macro-frame F_j and displayed at time T_{disp} (Figure 3).

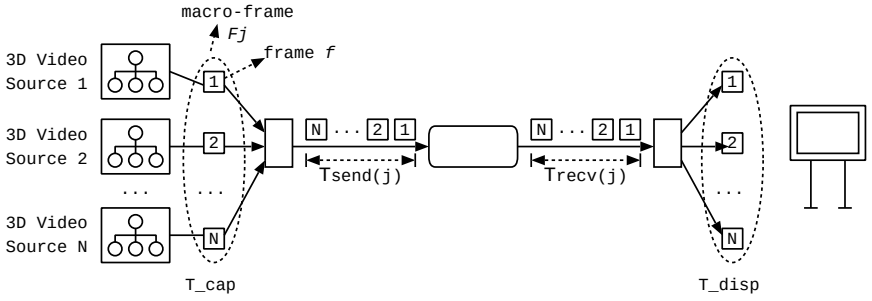


Fig. 3. Timing Model

To preserve this timing model, we will introduce an extensive coordination and synchronization service as well as end-to-end streaming control to meet the deadlines and skew requirements as discussed in later sections.

IV. End-to-End TEEVE Architecture

The ultimate goal of TEEVE is to provide a tele-immersive environment via COTS components and best-effort networks. To meet the challenge, we are proposing a flexible and cost-effective end-to-end cross-layer architecture that incorporates all participating devices for efficient resource utilization and quality of service. This section presents the roadmap of the overall architecture from the *user*, *layer* and *device* views (Figure 4).

[2]In our setup we consider half-circle organization of camera clusters.

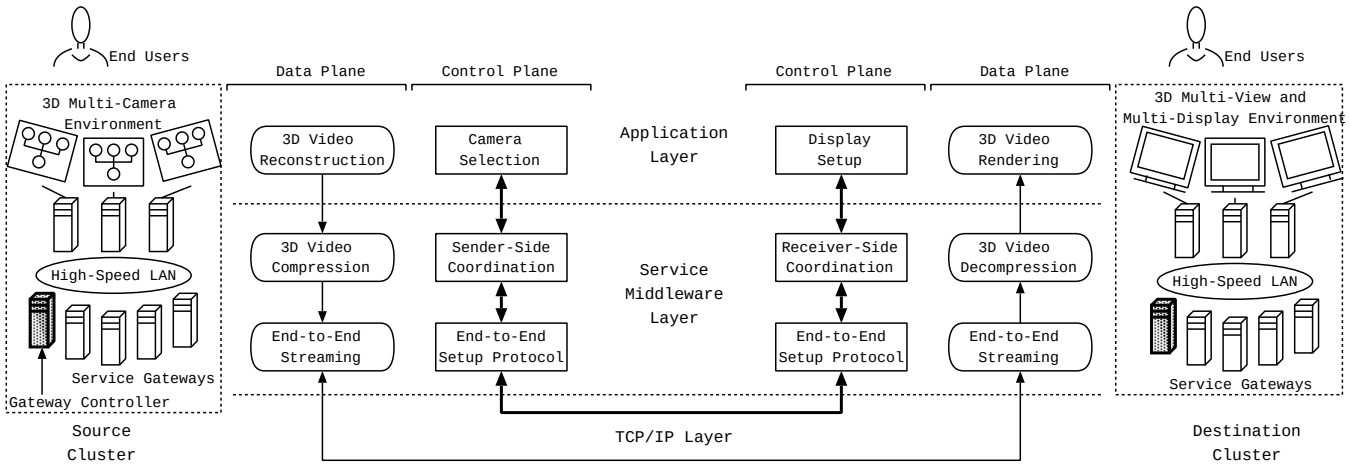


Fig. 4. TEEVE Architecture

1) User View: The user view represents the top-end of the tele-immersive environment, containing multiple 3D cameras and high definition displays organized in a semi-circle way. When the user enters the environment, his 3D representation is reconstructed and rendered in a virtual space shared by other remote users. The sense of immersion is greatly enhanced by 3D models as he interacts with the virtual space through different views and display layouts.

2) Layer View: The layer view of the cross-layer framework considers the peer-to-peer protocols and functions in each layer, and the interaction across the layers. The TEEVE architecture consists of three layers: the application layer, the *service middle layer* (SML), and the underlying TCP/IP layer. The design of the tele-immersion system concentrates on the top two layers. Furthermore, the TEEVE design uses a cross-layering cut along the control and data delivery functions of the two layers (as shown in Figure 4).

The task of application layer is to manipulate the raw 3D video data, control the high-quality 3D multi-camera/display environment for the end users, and send feedback horizontally across the three tiers of the application, and vertically across the application and SML layers. More specifically, in the data plane, it needs to perform real-time 3D reconstruction and rendering. In the control plane, it captures the response of the end user to carry out camera selection and display setup. For the cross-layer interaction, it needs to inform the lower layer of important system changes. For example, as the 3D scene is rendered by multiple 3D video streams a change of the view affects the relative importance of each stream. The information can be passed on to lower layer to affect the resource allocation for the underlying streaming control and bandwidth management. The service middleware layer provides the major infrastructure for the TEEVE environment. It is composed of a group of *service gateways* (SG) which are coordinated by the *gateway controller* to perform critical tasks of 3D video streaming, sender/receiver side coordination, and end-to-end feedback control.

3) Device View: The device view of TEEVE represents a highly concentrated area of capturing, processing, and displaying devices interconnected via different networks. In this paper, we consider 12 3D camera clusters as discussed in Section III-A. Two camera clusters are grouped together, which constitutes one major *scene view*. We have six scene views in our setting with different combinations of left/front/right and top/bottom. Each camera cluster or display connects to an edge-processing node, whose major function is 3D video reconstruction or rendering. The edge-processing nodes and service gateways are interconnected through high-speed LANs. The design also facilitates flexibility since the connection between edge-processing nodes and gateways can be dynamically configured. Finally, service gateways are connected to outer networks using general infrastructure.

In the next sections, we elaborate on the cross-layer design of the TEEVE architecture, taking into account the user, layer, and device view when considering the control and data delivery functions.

V. Control Plane

The control plane is responsible for optimizing resource utilization and QoS via end-to-end and cross-layer feedback channels. The problems addressed include end-to-end setup, multi-tier multi-stream coordination and synchronization, and multi-camera/display management.

A. End-to-End Setup Protocol

The end-to-end setup protocol is a control protocol between the *gateway controllers* of SML at both ends, exchanging the information about current bandwidth, applying streaming control, and supporting application control layer.

The first step of the gateway controllers is to initiate the tele-immersive environment at both ends. The gateway controller at the sender side begins the process with the collection of the local resource information about the capturing tier, including the number of service gateways and 3D camera clusters. It then contacts its peer at the other end via an *invite* message. The gateway controller at the receiver side responds back with information about the displaying tier. Following that, they negotiate and match the resources of each other.

In the second step, the local control and data paths are formed. In the capturing tier, the gateway controller configures the association between edge-processing nodes and service gateways. The association takes into account both load-balancing of gateways and the adjacency of cameras via the camera selection services. For the latter concern, as neighboring cameras are more likely to share similar viewing area, associating them with one service gateway will benefit inter-stream compression algorithm [4]. In our tele-immersive environment, we assign cameras of the same *scene view* to one service gateway.

The gateway controller at the receiver side has a similar problem of associating rendered streams and desired scene views with displays. The challenge is to decide the importance of each stream per display, and to dynamically

configure the association. Take the example of four video streams (Figure 5). Based on the current user view of the two displays, streams 1, 2 and 3 are the most important streams for display A, while stream 4 is optional.

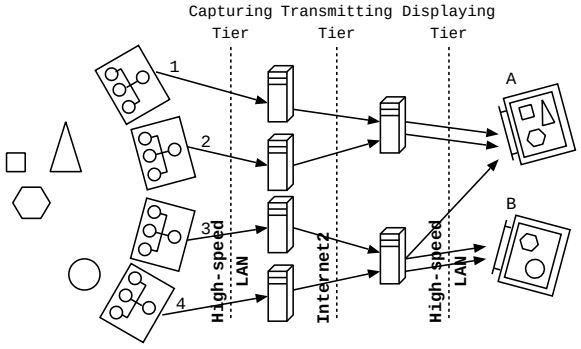


Fig. 5. An Ideal Mapping of Displays and Streams

During the third step, the gateway controllers broadcast the configuration to both capturing and displaying tiers, and service gateways to make them connected. Then the gateway controllers finalize the connection setup of the individual service gateways at both ends in a load-balanced fashion.

B. Multi-stream Coordination

Another important TEEVE control function is coordination and synchronization to transmit 3D macro-frame streams. In TEEVE, SML is responsible to transmit 3D video streams and since the macro-frames are being transmitted via TCP, a strong multi-streaming coordination is required in the transmission tier of the TEEVE application. TCP is chosen as the transport protocol due to its reliable and in-order delivery, the congestion control, the easiness of handling large packet sizes at the application layer, and the lower context switching overhead, which are well suited for streaming data of large volume. Even though the backoff and retransmission mechanisms seem to make it undesirable for streaming, it has been shown that TCP is widely used in commercial streaming systems [10]. In the future, we will compare the performance of TCP streaming with its counterparts of UDP based streaming such as selective retransmission under the tele-immersive context.

As pointed out in the literature ([4], [9], [13], [14], [15]), a multi-stream coordination mechanism is critical for streaming using multiple TCP flows, particularly where multiple streams have to share a common path. Without such mechanism, each TCP flow applies its congestion control independently. Under network congestion, some may detect congestion and backoff accordingly while other may not. The chaotic behavior of multiple TCP flows causes the fluctuation of bandwidth sharing, low throughput, and low degree of inter-stream synchronization.

Several schemes have been proposed for coordinating aggregated TCP traffic ([13], [14], [15]) from where we apply the basic concept of multiplexing TCP flows. However, our interest here is in designing a coordination

scheme at the SML layer. The problem is abstracted by the following model. Each sender (e.g. 3D camera cluster) needs to send a sequence of frames in order (e.g. $f_1^i, f_2^i, \ldots f_k^i$ from 3D camera i). There is a *counter* which has the current sequence number of the macro-frame F_j allowed to be sent. After every sender finishes sending the frame f_j^i ($i = 1, \ldots, n$) of the current sequence number j, the counter is incremented by 1. No sender can send a frame whose number is bigger than the current sequence number. A central gateway controller is employed to synchronize the senders. The controller waits until all senders are ready and sends a firing signal to them. The senders then send out one frame and wait for the next signal.

In this *basic scheme*, senders do not coordinate their sending time and each sends the frame as soon as it receives the synchronization signal. We set up an experiment using six senders and one receiver with a 100 Mbps link between the parties. The central controller sends out firing signal as soon as every sender finishes sending the current frame so as to push the limit of bandwidth. The frame size is 3,072 Kbits which is equal to the size of a 3D image frame. The performance parameters are plotted in Figure 6 which shows the overall throughput and the throughput of each individual stream.

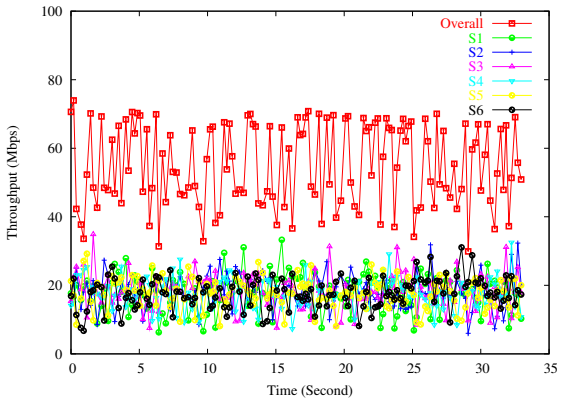


Fig. 6. Multi-streaming without Coordination

The results are consistent with those presented in [9]. The overall throughput is only 54.1 Mbps. The bandwidth measurement of each stream is 17.9 Mbps. We propose a coordination scheme based on the idea of spacing out the traffic of senders using *token ring* and satisfying the timing model discussed in Section III-B. To avoid congestion, the service gateways are organized into a ring topology. To start, the gateway controller passes on a *token* to the first gateway of the ring. When it receives the token, it sends out one frame and forwards the token to the next gateway. As the token travels along the ring, the transmission of each gateway is fired sequentially until it reaches back to the controller where it will be fired at the next instant.

The token-ring multi-stream coordination is used in each tier of TEEVE framework. The coordination of 3D camera clusters to send traffic over LAN and the coordination of service gateways to send the traffic over Internet2 comprise the sender-side coordination. The coordination of service gateways at the receiver-side to forward multiple streams to multi-view displays compries the receiver-side coordination.

C. Display Setup

As discussed in the end-to-end setup protocol, we need to associate the rendered 3D streams with the display space in the displaying tier. This task of display setup is a challenging problem, because there can be many scene views

in the 3D space to look at the objects and people, and the display space becomes a scarse resource very quickly. Traditional systems usually present a single view port that can be manipulated by the user, which requires the user to continuously interact with the system to adjust the view point. This is a tedious operation and may distract the user from the primary task of conferencing. In TEEVE, we employ an *innovative multi-window scheme* that associates multiple scene views with one or multiple displays based on personal preference. Next, we will describe first how the candidate scene views are selected, and then how users interact with the system to specify screen layouts of these scene views.

1) Selection of Scene Views: A set of default scene views (e.g. frontview) is first automatically computed using geometry of 3D camera locations, and then the user can easily adjust each of them. The scene views need to cover all view aspects of the scene yet should be simple to understand and control by the user. In the current system, all 3D cameras are located on a half circle facing the frontal view of the scene with equal spacing and facing the center of the scene, as shown in Figure 7 below.

Three parameters are necessary to generate view points that cover scene views: 3D center $C_0 = (x_0, y_0, z_0)$, the number of view points N_v, and the distance D between the view points and the 3D center (the radius of the circle). The 3D center C_0 can be automatically computed by averaging the 3D coordinates of all points in the scene. In our example (Figure 7), the number N_v is 6 by default and the distance D is 4 meters by default. To adjust a view point, a user can use keyboard to adapt its 3D coordinate and also its orientation.

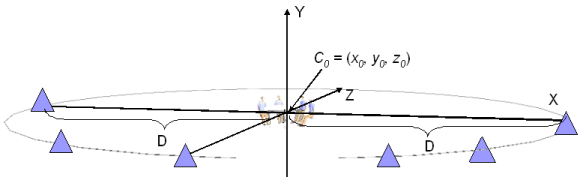


Fig. 7. Automatically Computed Scene Views

2) Selection of Screen Layouts: Three parameters determining how the scene views rendered at the view points are presented to the user: (1) the number of view points N_v, (2) the number of displays N_d, and (3) which view point is the *focus-view point* (FVP). Theoretically, any number of displays can be supported, but the current system limit is between 1 and 4, and by default N_d = 1. Based on the total number of view points to be presented and the number of available display devices, the system can determine how many views to render on each display device. A focus-view point is picked by the user so that it gets allocated more display space. To simplify the screen layout management, a set of default screen layouts is defined. For single display scenario, examples of default screen layouts for N_v between 1 and 6 are shown in Figure 8. For double display scenario, default screen layouts for some N_v values are shown in Figure 9 below.

The user can specify which view point is the FVP. For example, he may use the mouse to click on any of the non-FVP windows on any display, and the view point shown in that window will be exchanged with the current FVP. The information of the FVP is sent to the SML to differentiate 3D streams so that the video in the FVP

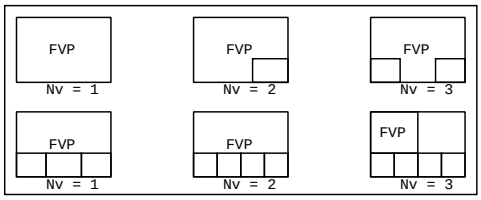


Fig. 8. Default Setup of Screen Layouts for $N_d = 1$

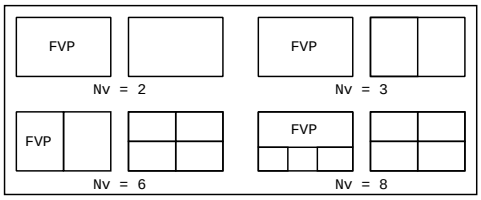


Fig. 9. Default Setup of Screen Layouts for $N_d = 2$

window is shown in high quality while other video windows are displayed in lower quality. Whenever the viewer picks a new view point to be shown in the FVP window, it will be shown in low quality at very low response time, and then the quality will pick up gradually as SML adapts the streaming policy accordingly.

VI. Data Plane

The challenges of the TEEVE data plane are the real-time compression and the multi-tier streaming functions for the 3D video data.

A. 3D Video Stream Compression

Due to the large volume of 3D data captured from the 3D camera clusters, service gateways need to perform compression.[3] This section addresses the issue of real-time compression of 3D video streams. At the resolution of 320×240, the size of one image is 3.1 Mbits with each pixel having 3 bytes for color and 2 bytes for depth. The basic bandwidth requirement is huge for streaming multiple 3D videos across the Internet2. The challenge of 3D compression as distinguished from 2D compression is the addition of the depth information. Here, we briefly present a hybrid compression scheme that first applies *color reduction* to compress the color information, followed by *background pixel removal* and *zlib* compression.[4]

1) Color Reduction: The color reduction is one of the popular image compression techniques [17], [18] to reduce the bits for colors (e.g. from 24 bits to 8 bits per pixel) while maintaining pretty high visual quality. We adapt one color reduction algorithm based on ImageMagick [17] for the tele-immersive application, which contains three phases: *color classification, color reduction,* and *color assignment*.

Color classification builds a color description tree for the image in RGB color space. The root of the tree represents the entire space from $(0,0,0)$ to $(C_{max}, C_{max}, C_{max})$ where $C_{max} = 255$. Each lower level is generated by subdividing the cube of one node into eight smaller cubes of equal size. For each pixel in the image, classification

[3]Note that, the existing schemes such as MPEG, JPEG and H.263 cannot be applied because the video not only include RGB information, but also depth information which would be lost if using standard schemes. Furthermore, we need the depth information in the displaying tier to render the macro-frames into full-scale 3D frames.

[4]More details can be found in [16].

scans downward from the root of the color description tree. At each level, it identifies the single node which represents a cube containing the color of the pixel and updates the statistics of such node.

Color reduction collapses the color description tree until the number it represents is at most the number of colors desired. The goal is to minimize the numerical discrepancies between the original colors and quantized colors. For this, color reduction repeatedly prunes the tree. On any given iteration over the tree, it selects those nodes whose quantization error is minimal for pruning and merges their color statistics upward.

Finally, color assignment generates the output image from the pruned tree. It defines the *color map* of the output image and sets the color of each pixel via indexing into the color map. For the example of 24-bit to 8-bit color reduction, the output image contains the color map of 256 bytes and an index array of pixels which is one third the size of the original pixel array, achieving a total compression ratio close to 3.

2) Background Pixel Removal: For tele-immersive application, the foreground pixels carry the most useful image information. The background pixels are usually blank after the initial background segmentation and can be entirely eliminated (e.g. Figure 10). To perform background pixel removal, a *bitmap* is generated for each frame and attached at the frame head showing for each pixel whether it belongs to the background or foreground. Since each entry only consumes 1 bit, the additional overhead is acceptable. In a typical image captured from the tele-immersive environment, the background pixels account for around 70% of total pixels, yielding a compression ratio over 2.3.

3) zlib Compression: After the previous two steps, the color information is shrunk to almost one seventh of its original size. As the last step, the *zlib* [19] compression, a powerful, generic and openly available compression tool, is applied to further improve the compression ratio. The decompression follows similar reverse steps except that color reduction is much simpler. Once the bitmap, the color map, and the color index array are restored, the original color information can be easily recovered.

4) Practical Issues: Color reduction is usually regarded as an expensive operation, which may impact its real-time performance. However, since the basic color layout of all image frames contained in a tele-immersion session does not change dramatically (e.g. the colors of hair, face and clothes), the full-fledged color reduction algorithm does not have to be performed for each frame. The first frame of the session has to go through all three phases. Once the color description tree has been set up and properly pruned. The follow-up frames can be compressed using only the assignment phase. Since most computing-intensive operations of the algorithm happen during the classification and reduction phases, much computing overhead is saved.

5) Evaluation: The compression scheme is written in C++ and tested on Dell Precision 450 running Fedora Core 2 system. For evaluation, we use one of the actual 3D video streams pre-recorded in the tele-immersive environment showing a person and his physical movements (Figure 10). Currently, we are using an image resolution of 320×240.

The video stream contains 612 depth frames with a total size of 235 MBytes.

Table I shows the compression time in unit of one millisecond (ms), while Table II shows the decompression time. For compression, the most expensive time cost is color classification and reduction with an average around 35 ms. However, as mentioned earlier it is reasonable to assume a stable color layout in tele-conferecing environment such that the classification and reduction phases only need to be performed for the first frame of the session or every n frames. For a stable color layout, n could take a pretty large value (e.g. $n = 100$ assuming an average recalculation period of 10 seconds and a frame rate of 10 frames per second). Therefore, the average compression time can be taken as around 10 ms (i.e., the sum of last three steps).

TABLE I
COMPRESSION TIME

	min	avg	max
classification	0.416	7.11	14.4
reduction	0.001	27.6	80.9
assignment	0.878	4.19	9.29
background removal	0.641	0.708	1.54
zlib compression	2.66	4.93	11.1

TABLE II
DECOMPRESSOIN TIME

	min	avg	max
decompression	1.36	1.72	3.99

Table III gives the compression ratio achieved for the 612 3D images frames. The performance of compression is very impressive (26.7), which implies that on average the overall frame size can be shrunk from 384 KBytes to below 15 KBytes. This excellent compression reflects the large amount of spatial redundancy in the 3D image, which consists of a person moving against an empty background.

TABLE III
COMPRESSION RATIO

	min	avg	max
color ratio	10.1	14.2	22.2
overall ratio	14.9	26.7	358

TABLE IV
VISUAL FIDELITY

	min	avg	max
color	45.9	51.5	74.3

The visual fidelity of the color information after decompression is measured using PSNR (in dB). The results are given in Table IV, which indicates that the compression scheme has pretty high compression quality. We also visually judge the image quality by comparing two similar frames before and after data compression as in Figure 10 and 11. No quality degradation is observed.

The experimental evaluation shows that this compression scheme is well suited to our problem domain, giving good compression ratios as well as real-time performance and visual quality. Although color reduction is usually considered a time-consuming operation, under the context of tele-immersive environments most of the computing

Fig. 10. Visual Quality before Compression

Fig. 11. Visual Quality after Compression

overhead can be eliminated to achieve very high performance for 3D video compression.

B. End-to-End Multi-Tier Streaming Protocols

The end-to-end streaming protocols address the challenges of bandwidth management and rate adaptation, especially for streaming of 3D macro-frame video data where resource demand is much higher than for any 2D streaming. Since the capturing and displaying tiers use high-speed LAN and the bottleneck is in the transmitting tier, the main focus is the streaming in the transmitting tier. In this section, we discuss end-to-end streaming functions that need to be in place including bandwidth estimation, bandwidth allocation, and rate adaptation.

1) Bandwidth Estimation: The first step of the end-to-end streaming protocol is to estimate available bandwidth. However, bandwidth estimation turns out to be a very complex problem and we present here only a coarse estimation scheme. The clocks of service gateways at the sender side are accurately synchronized using the Network Time Protocol (NTP). As the token passes along the ring, the service gateways attach the information abount sending time, round trip delay and data size of recent packets to the token. When the token arrives at the gateway controller, the information is collected to compute the overall throughput.[5]

2) Bandwidth Allocation: Under bandwidth constraints when the available bandwidth is not sufficient to support all the 3D streams at the minimum frame rate, the gateway controller needs to decide the bandwidth allocation for each stream. We adopt a view-based allocation scheme by pre-defined priorities and the dynamic view changes as discussed in [20]. As an example, the 3D cameras of the front scene view are generally more important than those on the side view and are given higher priority. During runtime, the user may select different views, which affects the bandwidth allocation as well.

After the bandwidth allocation is calculated at the gateway controller, it is attached to the coordinating token and conveyed to the service gateways. When a service gateway receives the token, it allocates the bandwidth to each

[5]As mentioned earlier, the bandwidth estimation is performed in the transmit tier over Internet2.

3D stream according to the allocation schedule described in the token.

3) Rate Adaptation: The rate adaptation of 3D macro-frame streaming presents a much more interesting problem than the case of 2D streaming, where we can apply different adaptation schemes in a two-dimensional space (Figure 12). The simplest scheme is to turn off less important streams or to interleave among equally important streams. If a multi-view display is used, we assign less bandwidth to the less important streams (e.g. a side view). However, we need to study the rendering quality and how this affects the immersive feeling of the participants.

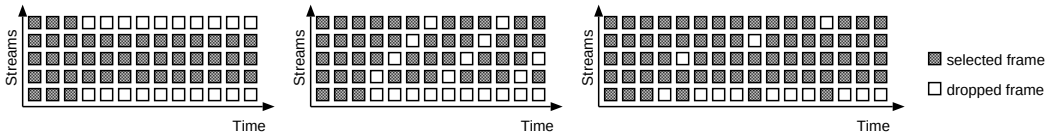


Fig. 12. Simple Rate Adaptations

VII. Experimental Results

In this section, we present the experimental results of the 3D tele-immersion streaming via real networking testbed. The first suite of experiments is carried out under well-controlled environment. We want to analyze how the performance of 3D macro-frame streaming coordination is influenced by various coordinating factors. In the second suite of experiments, we want to analyze performance of streaming real 3D videos over Internet2.

A. Experimental Environment

We have implemented the 3D video capturer, multi-view 3D renderer, service gateway, and gateway controller. All programs are written using C or C++. The 3D capturer and renderer are developed for Windows XP platform while service gateway and gateway controller have both versions of Windows XP and Fedora Core 2.

To analyze the cost of individual algorithms and protocols, we have established two networking testbeds: (a) local testbed where we capture/store 3D videos, forward them to local service gateways, and send them over 100Mbps link to the renderer with single view display, and (b) wide area testbed from UIUC to UC Berkeley.

B. Metrics

We are interested in measuring not only the level of synchrony but also the overall throughput of sending each macro-frame (*frame ensemble* in [4]). We sequence macro-frames from 1 to infinity and number the capturing devices from 1 to N (each cooresponds to one stream). Then each frame is uniquely identified as f_j^i where i represents the index of capturing device ($i \in [1..N]$) and j represents the sequence of marco-frame ($j \in [1..\infty]$).

For frame f_j^i, we define $SIZE_j^i$ as its size, $SEND_j^i$ its sending time, and $RECV_j^i$ its receiving time. The *overall throughput* (OT) of macro-frame j is given in (1). The *completion time interval* ($T_{recv(j)}$) for macro-frame j is given

in (2), similar to the *completion time* as in [4]. The *individual throughput* (IT) of stream i is given in (3).

$$OT_j = \frac{\Sigma_{i=1}^{N} SIZE_j^i}{max_i(RECV_j^i) - min_i(SEND_j^i)} \tag{1}$$

$$T_{recv(j)} = max_i(RECV_j^i) - min_i(RECV_j^i) \tag{2}$$

$$IT_j^i = \frac{SIZE_j^i}{RECV_j^i - SEND_j^i} \tag{3}$$

C. Experiments of Local Testbed

The experimental parameters of our local testbed are listed in Table 13. We measure the performance of the multi-stream coordination, and compare the basic scheme with the token-ring scheme under two scenarios: (1) streaming as fast as possible and (2) streaming with fixed sending rate.

Fig. 13. Experimental Parameters of Local Testbed

number of sender gateways (NSG)	6
number of receiver gateways (NRG)	1
number of 3D streams (NS)	6
size of raw image	3.072 Mbits

Fig. 14. Experimental Parameters of Remote Testbed

number of sender gateways (NSG)	2
number of receiver gateways (NRG)	2
number of 3D streams (NS)	12
frame rate	4 fps

1) Scenario 1: In this scenario, we do not apply rate control. As soon as the gateway controller gets the token, it forwards it to either all the gateways (basic scheme) or the first one on the ring (token-ring scheme). The performance of the basic scheme is shown in Section V-B. For the token-ring scheme, each gateway forwards the token as soon as it finishes sending its current frame. The performance parameters are plotted in Figure 15.

As can be seen, the average overall thoughput is improved to 75.9 Mbps (basic scheme: 54.1 Mbps) while the average individual throughput of each stream is 76.9 Mbps. The individual throughput measurement indicates a less degree of congestion which could be further avoided with the introdcution of *hold after transmit* (HAT) time. That is, after the sender sends out a frame it keeps the token for a short time period before forwarding it to the next sender. In the following experiments, we select the HAT time from 1 ms to 10 ms and summarize the results in Figure 16. Figure 16 confirms that with larger HAT time the throughput does not increase, and the maximum reaches between 1 and 2 ms, showing a 43.9% improvement over the base case (77.9 Mbps vs 54.1 Mbps).

2) Scenario 2: In the previous scenario, the gateway controller passes the token without applying extra HAT delay to push the bandwidth limit and the network utilization is very high. Generally, the goal of streaming is to apply rate control instead of sending as fast as possible and we are also interested in the performance of token-ring scheme in the situation of lower channel contention. Note that, it is convenient for the gateway controller to measure the token travel time and keep the token for certain amount of time to control the frame sending rate.

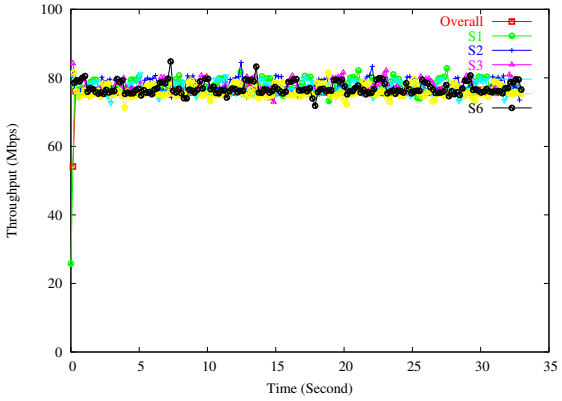


Fig. 15. Multi-streaming with Token-ring

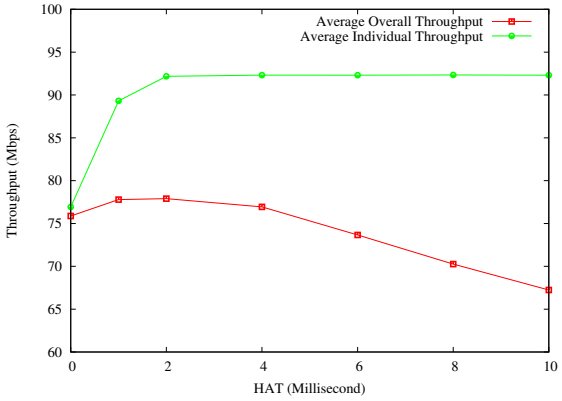
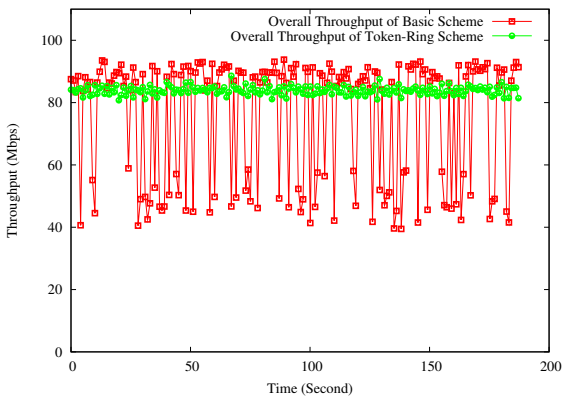
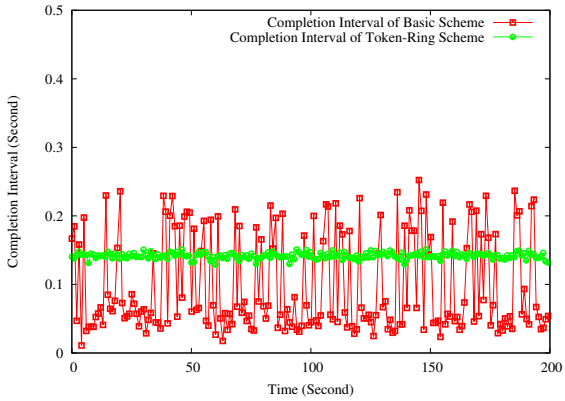


Fig. 16. Performance of Various HAT

The experimental results in Figure 17 compares the overall throughput and receiving interval in both basic and token-ring schemes ($NS = 5$, $FR = 1$ fps, $HAT = 0$). The total number of frames sent by each stream is 200, which is sufficient to stabilize the results. Each experiment has been run for several rounds to confirm the output.



(a) Overall Throughput



(b) Completion Interval, $T_{recv(j)}$

Fig. 17. Impact of Fixed Sending Rate

After the control of sending rate is applied, the channel contention is reduced. This helps to increase the throughput of the basic scheme. However, the token-ring scheme provides a much smoother throughput delivery which is desirable for tele-immersive video. In the local 100 Mbps testbed, we can support up to 6 uncompressed video streams with a frame rate greater than 4 fps. The following experiments will show that we can achieve a similar frame rate with up to 12 compressed video streams.

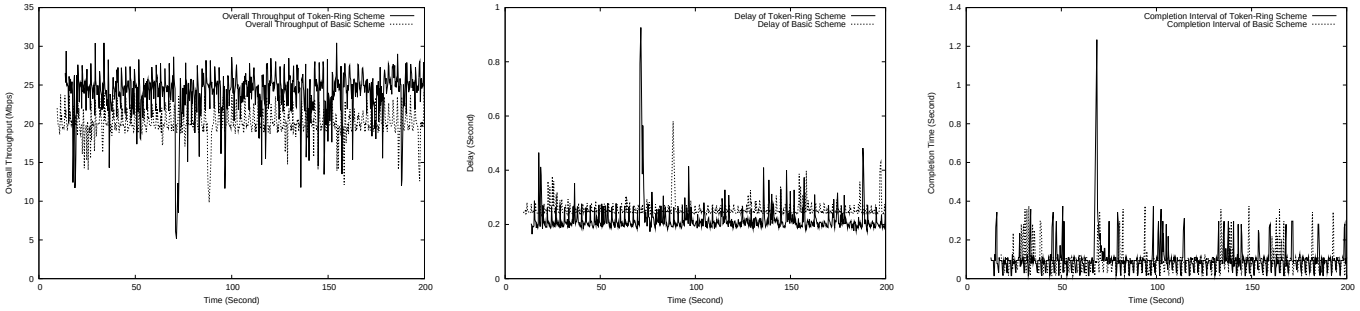
D. Experiments of Remote Testbed

To our best knowledge, we are the first one to present the results of tele-immersion streaming across the Internet2 using 12 3D video streams. So far, the only tele-immersion streaming test across the Internet is presented by [1]

but the 3D video streams are rendered locally with only the 2D video streams being sent, and the number of 3D streams is 5. The experimental parameters of our remote testbeds are listed in Table 14.

We assume the parallel processing of gateways while the token is passed along the ring. That is, as soon as the gateway sends the current frame and forsakes the token it can pick up the next frame to start compression. For the token-ring scheme, we set $HAT = 1$ ms. We use our real-time compression algorithm for data compression (Section VI-A). The experimental results are listed in Figure 18. The macro-frame delay is the end-to-end delay of macro-frame j as given in (4).

$$DELAY_j = max_i(RECV_j^i) - min_i(SEND_j^i) \tag{4}$$



(a) Overall Throughput (b) Marco-Frame Delay (c) Completion Interval, $T_{recv(j)}$

Fig. 18. Streaming Performance of Remote Testbed

The figure shows that the token-ring scheme has improvement of the overall throughput over the basic scheme. However, due to the variation of Internet traffic the improvement is less dramatic as compared with the more controllable local testbed environment.

VIII. Conclusion

We have presented TEEVE, a tele-immersive framework, that integrates the tele-immersive 3D camera array edges with the general purpose computing and communication infrastructure in a QoS fashion. Using careful coordination and synchronization among service gateways, real-time compression, and cross-layer multi-tier streaming protocols, we have shown that (a) we can deliver more than 4 macro-frames per second (consisting of six uncompressed 3D video streams) across LAN to the service gateways, and (b) we can sustain this frame rate of compressed macro-frames with each macro-frame containing 12 compressed 3D video frames across the wide area network up to the display. These results are starting to present a real opportunity for broader audience such as artists, social scientists, and others.

REFERENCES

[1] H. Baker, N. Bhatti, D. Tanguay, I. Sobel, D. Gelb, M. Goss, W. Culbertson, and T. Malzbender, "Understanding performance in coliseum, an immersive videoconferencing system," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 1, 2005.

[2] M. Chen, "Design of a virtual auditorium," *MULTIMEDIA '01: Proceedings of the 9th annual ACM international conference on Multimedia*, 2001.

[3] L. Gharai, C. Perkins, C. Riley, and A. Mankin, "Large scale video conferencing: A digital amphitheater," in *8th International Conference on Distributed Multimedia Systems*, 2002.

[4] D. E. Ott and K. Mayer-Patel, "Coordinated multi-streaming for 3d tele-immersion," in *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*. New York, NY, USA: ACM Press, 2004, pp. 596–603.

[5] S. Shi, L. Wang, K. Calvert, and J. Griffioen, "A multi-path routing service for immersive environments," in *Workshop on Grids and Advanced Networks, in conjunction with CCGrid 2004*, 2004.

[6] O. Schreer, N. Brandenburg, S. Askar, and E. Trucco, "A virtual 3d video-conferencing system providing semi-immersive telepresence: A real-time solution in hardware and software," in *International Conference on eWork and eBusiness*, 2001.

[7] T. M. Project, "http://www.netlab.uky.edu/theme.html," 2001.

[8] S.-U. Kum, K. Mayer-Patel, and H. Fuchs, "Real-time compression for dynamic 3d environments," in *MULTIMEDIA '03: Proceedings of the 11th annual ACM international conference on Multimedia*, 2003.

[9] H. Towles, S.-U. Kum, T. Sparks, S. Sinha, S. Larsen, and N. Beddes, "Transport and rendering challenges for multi-stream 3d tele-immersion data," in *NSF Lake Tahoe Workshop on Collaborative Virtual Reality and Visualization (CVRV'03)*, October 2003.

[10] B. Wang, J. Kurose, P. Shenoy, and D. Towsley, "Multimedia streaming via tcp: an analytic performance study," in *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*. New York, NY, USA: ACM Press, 2004, pp. 908–915.

[11] J. Mulligan and K. Daniilidis, "Real time trinocular stereo for tele-immersion," in *International Conference on Image Processing*, 2001, pp. III: 959–962.

[12] J. Mulligan, V. Isler, and K. Daniilidis, "Trinocular stereo: A real-time algorithm and its evaluation," *International Journal of Computer Vision*, vol. 47, no. 1-3, pp. 51–61, April 2002.

[13] P. Pradhan, T. cker Chiueh, and A. Neogi, "Aggregate tcp congestion control using multiple netowrk probing," in *Proceedings of the 20th International Conference on Distributed Computing Systems (ICDCS 2000)*, April 2000.

[14] P. Tinnakornsrisuphap, R. Agrawal, and W. chun Feng, "Rate-adjustment algorithm for aggregate tcp congestion control," in *Los Alamos Unclassified Report [LA-UR 00-4220]*, 2001.

[15] H. T. Kung and S. Y. Wangap, "Tcp trunking: Design, implementation and performance," in *Proceedings of the 7th International Conference on Network Protocols (ICNP'99)*, 1999.

[16] Z. Yang, Y. Cui, Z. Anwar, R. Bocchino, N. Kiyanclar, K. Nahrstedt, R. H. Campbell, and W. Yurcik, "Real-time 3d video compression for tele-immersive environments," University of Illinois at Urbana-Champaign, Tech. Rep. UIUCDCS-R-2005-2620, August 2005.

[17] "Imagemagick, http://www.imagemagick.org/script/quantize.php," http://www.imagemagick.org/script/quantize.php.

[18] "Color reduction, http://www.catenary.com/appnotes/colred.html," http://www.catenary.com/appnotes/colred.html.

[19] "Zlib 1.2.2, http://www.zlib.net," http://www.zlib.net.

[20] Z. Yang and K. Nahrstedt, "A bandwidth management framework for wireless camera array," in *International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'05)*, 2005.