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
{zyang2, klara}@uiuc.edu

Sang-hack Jung, Ruzena Bajscy
University of California at Berkeley
Department of EECS
253 Cory Hall, Berkeley, CA 94720
{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.

1. 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, 8, 12]). 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 middleware 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 teleimmersive streaming experiment. Our next step will focus on performing the user study. Section 2 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 3. In Section 4, 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 middleware layers, we discuss TEEVE solutions across two major communication planes, the control plane and the data transport plane. In Section 5, we introduce functions, services and protocols to setup, coordinate and adapt TEEVE. Section 6 presents the multi-tier end-to-end streaming protocol. In Section 7, we validate TEEVE performing extensive LAN/WAN experiments. Section 8 concludes by discussing our findings with the TEEVE system.

2. Related Work

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

Most video conferencing systems fall into the first category, such as PolyCom, Microsoft NetMeeting and WebEx, which aim to provide point-point or limited multipoint 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.

Solutions in the second category aim to provide the telepresence experience beyond single 2D views. In [2] and [3], participants of video conference are grouped and tiled into synthetic environments. Coliseum [1] is a desktop-based immersive conferencing system, where the 3D view of a user is captured by multiple cameras, extracted from background, then reconstructed and embedded into the virtual environment. 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 [11], MetaVerse [10][12], and the National Tele-Immersion Initiative [4][8][14]. 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. [15] 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 [12], an overlay-based multipath 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 [4], 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 aimed for high quality immersive experience while relying on general purpose computing and networking infrastructures.

3. TEEVE Application Model

The TEEVE application is modeled as a distributed multi-tier application (Figure 1). The first is the *capturing tier* that consists of 3D camera clusters with each cluster reconstructing one 3D video stream. The 3D streams are then

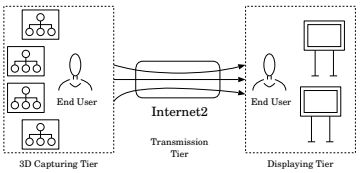


Figure 1: TEEVE Model

Figure 2: Environment

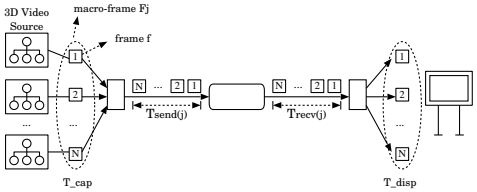


Figure 3: Timing Model

forwarded over LAN to service nodes for compression. The second is the *transmission tier* that streams the compressed video over Internet2. The third is the *displaying tier* which decompresses and renders the received 3D streams into an immersive video and forwards it over LAN to multiple displays. Figure 2 presents a pictorial view of tele-immersive environment with 3D camera clusters and displays. Each tier of the TEEVE framework deploys algorithm, service and protocol that require clear and sound application data and timing models.

3.1. TEEVE Data Model

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

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

The 3D data is created by a *cluster* of four cameras to capture color and depth from each viewpoint. The synchronization of cameras is achieved via hotwires. More specifically, three b/w cameras compute depths by adopting a trinocular stereo algorithm ([6, 7]), and a color camera extracts appearance from the corresponding viewpoint. The data returned by each cluster consists of an array of *XYZRGB* information for each pixel (3 bytes RGB and 2 bytes XYZ). We aim to reconstruct 3D frames at the rate of 10 fps. As each frame contains 320×240 pixels, one 3D video stream, uncompressed, demands 30.72 Mbps.

Integrated Data Model. The capturing tier consists of N equal 3D camera clusters organized in a half-circle (or circle) and each captures one 3D image at a time T_{cap}. At T_{cap} the capturing tier must have N 3D reconstructed frames constituting a *macro-frame*. This macro-frame F consists of $(f^1, ..., f^N)$ single frames from the individual clusters. Hence, the tele-immersive application yields a stream of macro-frames that are captured at the *macro-frame-rate* 10 fps (or a *macro-frame-period* of 100 ms).

3.2. Timing Model

Although one macro-frame F_j consists of N 3D frames (f^i), they cannot be transmitted at the same time due to possibly strong congestion. Hence, the individual frames f^i 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 then be rendered into the macro-frame and displayed at time T_{disp} (Figure 3).

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.

4. End-to-End TEEVE Architecture

To meet the challenge of providing a tele-immersive environment via COTS components and best-effort networks, we propose a flexible and cost-effective end-to-end cross-layer architecture that incorporates various devices for efficient resource utilization and quality of service. This section presents the overall architecture from the *user*, *layer* and *device* views (Figure 4).

User View. The user view represents the top-end of TEEVE, containing multiple 3D cameras and displays. 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 via 3D models as he interacts with the virtual space through different views and display layouts.

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 them. There are three layers: the application layer, the *service middle layer* (SML), and the underlying TCP/IP layer. The design of TEEVE concentrates on the top two layers and applies a cross-layering cut along the control and data functions (Figure 4).

The task of application layer is to manipulate the raw 3D video, control the high-quality multi-camera/display envi-

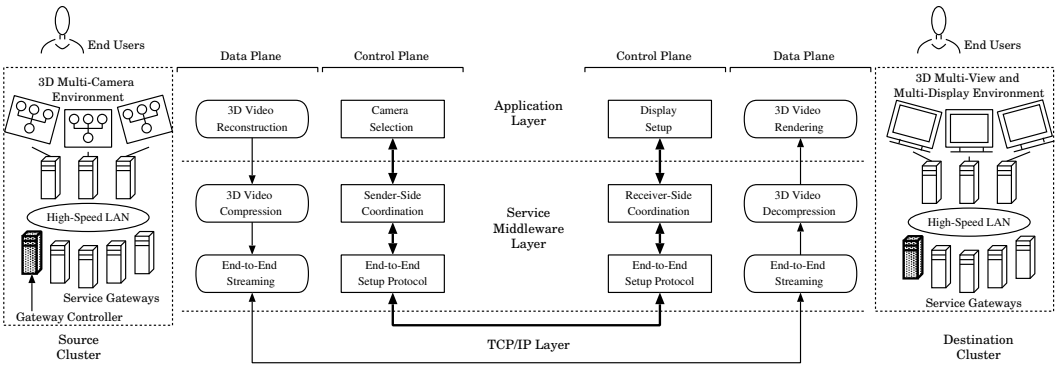


Figure 4: TEEVE Architecture

ronment, and send feedback horizontally across the three tiers, and vertically across the application and SML layers. In the data plane, it performs real-time 3D reconstruction and rendering. In the control plane, it captures the user response to carry out camera selection and display setup. For cross-layer interaction, it needs to inform the lower layer of important system changes. For example, a user view change affects the relative importance of each 3D stream. The lower layer can use the information to tune resource allocation for streaming and bandwidth management. The SML provides the major TEEVE infrastructure. It is composed of *service gateways* (SG) coordinated by the *gateway controller* (GC) to perform 3D streaming, sender/receiver coordination, and end-to-end feedback control.

Device View. The device view represents the capturing, processing, and displaying devices interconnected via different networks. Two camera clusters are grouped together, which constitutes a *scene view*. There are six scene views with different combinations of left/front/right and top/bottom. Each camera cluster or display connects to an edge-processing node for reconstruction or rendering. The edge-processing nodes and service gateways are interconnected through high-speed LANs. The design also facilitates flexibility as the connection can be dynamically configured. Finally, service gateways are connected to outer networks using general infrastructure.

Next, we elaborate on the cross-layer design of TEEVE architecture, taking into account the user, layer, and device views when considering the control and data functions.

5. 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.

5.1. End-to-End Setup Protocol

The end-to-end setup protocol between the GCs at both ends carries out the tasks of exchanging the current resource information, applying streaming control, and supporting application control layer.

The first step is to initiate the tele-immersive environment. The GC at the sender side informs the receiver side of the local resource about the capturing tier, including service gateways and camera clusters via an *invite* message. The receiver GC responds with information about the displaying tier. Then, they negotiate and match the resources.

Next, the local control and data paths are formed. For the capturing tier, the GC configures the association between edge-processing nodes and service gateways, taking into account both load-balancing of gateways and the adjacency of cameras. For the latter concern, as neighboring cameras are more likely to share similar viewing area, associating them with one service gateway will benefit processing like inter-stream compression. The receiver GC has similar problem of associating rendered streams and desired 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, streams 1, 2 and 3 are more important for display A, while stream 4 is optional.

Finally, the gateway controllers send 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.

5.2. Multi-stream Coordination

The service middleware layer is responsible to transmit 3D macro-frame streams. For the transport protocol, TCP is chosen due to its reliable and in-order delivery, the congestion control, the easiness of handling large packets, and the lower context switching overhead, which are well suited for streaming data of large volume. Although 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 [15]. In the future, we will compare the performance of TCP streaming with its counterparts of UDP based streaming such as selective retrans-

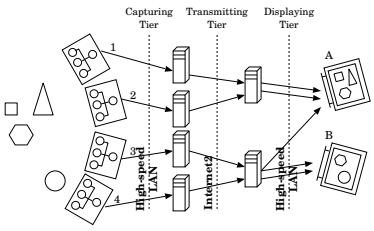


Figure 5: Stream/Display Mapping

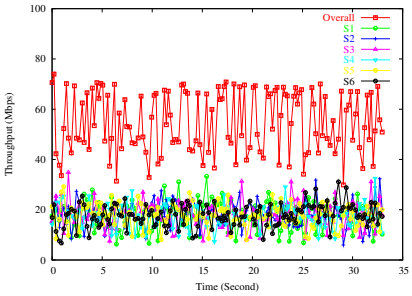


Figure 6: Streaming w/o Coordination

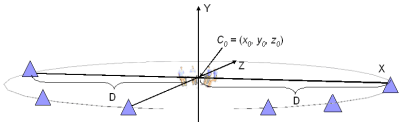


Figure 7: Computed Views

mission under the tele-immersive context.

As pointed out in the literature ([8], [14], [9], [13], [5]), a coordination mechanism is critical for streaming using multiple TCP flows. Without such mechanism, each TCP flow applies its congestion control independently. Under constraints, some may detect congestion and backoff accordingly while other may not. The chaotic behavior of 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 ([9], [13], [5]) from where we apply the basic concept of multiplexing TCP flows. The problem is abstracted by the following model. Each sender i sends a sequence of frames (i.e., $f_1^i, f_2^i, ...f_k^i$). There is a *counter* which holds the current sequence number of the macro-frame F_j to be sent. After every sender finishes sending the frame f_j^i of the current sequence number j, the counter is incremented. No sender can send a frame whose number is bigger than the current sequence number. A central 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 signal. We set up an experiment using six senders and one receiver within a 100 Mbps LAN. The central controller sends out firing signal as soon as every sender finishes sending the current frame. The frame size is 3,072 Kbits which is equal to that of the image frame. The performance parameters are plotted in Figure 6 which shows the overall throughput and the throughput of each individual stream.

The results are consistent with [14]. 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 using *token ring* and satisfying the timing model (Section 3.2). To avoid congestion, the SGs are organized into a ring topology. The GC passes on a *token* to the first gateway. The gateway then 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.

5.3. Display Setup

As discussed in Section 5.1, we need to associate the rendered 3D streams with displays in the displaying tier. The task of display setup is challenging as there can be many scene views in 3D space, and the display space becomes a scarse resource. Traditional systems usually present a single view port that can be manipulated by the user, but 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. We will describe how the candidate scene views are selected, and then how users interact with the system to specify screen layouts.

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. Currently, all 3D cameras are located on a half-circle with equal spacing and facing the front and center of the scene (Figure 7). Three parameters are necessary to generate view points: 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 3D center C_0 can be automatically computed by averaging the 3D coordinates of all points in the scene. To adjust a view point, a user can use keyboard to adapt its 3D coordinate and orientation.

Selection of Screen Layouts. Three parameters determining how the scene views rendered are presented to the user: the number of view points N_v, the number of displays N_d, and which view point is the *focus-view point* (FVP). Based on the total number of view points 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 layout management, a set of default screen layouts are defined. Examples of default screen layouts for N_v between 1 and 6 are shown in Figure 8.

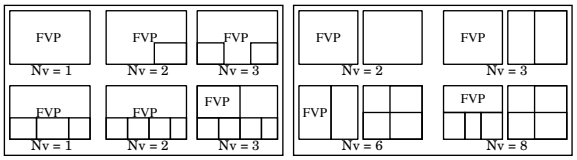


Figure 8: Screen Layouts for $N_d = 1$ and $N_d = 2$

The user can specify which view point is the FVP. For example, he may use the mouse to click on the non-FVP windows, 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 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.

6. Data Plane

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

6.1. 3D Video Stream Compression

The bandwidth requirement is huge for streaming multiple 3D video streams (Section 3) to call for the real-time compression of 3D video. As distinguished from 2D compression, existing schemes such as MPEG, JPEG and H.263 cannot be applied directly because the video contains both RGB and depth. Here, we briefly present a hybrid compression scheme that first applies *color reduction* to compress the color, followed by *zlib* compression (more in [16]).

The color reduction reduces the color bits while maintaining visual quality. We adapt an algorithm based on ImageMagick (www.imagemagick.org), which has three phases: *classification*, *reduction*, and *assignment*. The goal is to create a suitable *color map* to minimize the discrepancies between the original and quantized colors via the *color description tree*. The output image sets the color of each pixel via indexing into the color map. The final step applies *zlib* compression, a powerful, generic and open compression tool, to further compress color (reduced) and depth.

The color reduction is as an expensive operation, especially the classification and reduction phases. However, since the color layout in a session does not change dramatically (e.g. the colors of hair, face and clothes), the full-fledged algorithm does not have to be performed for each frame. The first frame needs to go through all three phases. Once the color description tree has been set up. The follow-up frames can be compressed using only the assignment phase to save computational cost.

The evaluation based on actual 3D videos shows that this compression scheme is well suited to the problem domain, giving good compression ratios (avg. > 26), real-time performance and visual quality (more in [16]). The average time cost of classification and reduction is 35 ms per frame. However, based on the former assumption the two phases only need to be performed for the first frame or every n frames, where n could take a pretty large value. Therefore, the average compression time would be close to the sum of color assignment and zlib compression ($\approx$ 10 ms/frame).

6.2. End-to-End Multi-Tier Streaming

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

Bandwidth Estimation. The estimation of available bandwidth is critical for streaming protocol. We present a coarse estimation scheme. The clocks of sender SGs are accurately synchronized using the Network Time Protocol (NTP). As the token passes along the ring, the service gateways attach the timing information and data size of recent packets to the token. When the token arrives at the GC, the information is collected to compute the overall throughput.

Bandwidth Allocation. Under bandwidth constraints, the gateway controller needs to decide the bandwidth allocation for each stream. We adopt a view-based allocation scheme by priorities and the dynamic view changes similar to [18]. For example, the front scene views 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. The bandwidth allocation is attached to the token and conveyed to the SGs. When it receives the token, it allocates the bandwidth to each 3D stream according to the allocation schedule.

Rate Adaptation. The rate adaptation of 3D macro-frames presents a more interesting problem than 2D streaming, where we can apply different adaptation schemes in a two-dimensional space. For example, we can turn off less important streams or to interleave among equally important streams (Figure 9). We need to study the rendering quality and how this affects the immersive experience of users.

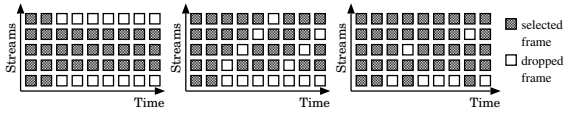


Figure 9: Simple Rate Adaptations

7. Experimental Results

This section presents the experimental results of TEEVE. The first suite of experiments is carried out under well-controlled environment to analyze how the performance of

3D macro-frame streaming is influenced by various coordinating factors. In the second suite, we analyze the performance of streaming real 3D videos over Internet2.

7.1. Experimental Environment

We have implemented the 3D video capturer (PointGrey Research), multi-view renderer (OpenGL), service gateway, and controller using C or C++ for both Windows and Linux platforms (Dell Precision 470 with 1GB memory). We have also established two networking testbeds: (a) local testbed within 100Mbps LAN, and (b) wide area testbed over Internet2 from UIUC to UC Berkeley.

7.2. Metrics

We are interested in measuring not only the level of synchrony but also the overall throughput of sending macro-frames (*frame ensemble* as in [8]). We sequence macro-frames from 1 to infinity and number the capturing devices from 1 to N. Then each frame is uniquely identified as f_j^i where $i \in [1..N]$ and $j \in [1..\infty]$.

For frame f_j^i, denote $SIZE_j^i$ as its size, $SEND_j^i$ its sending time, and $RECV_j^i$ its receiving time. The *overall throughput* is: $OT_j = \frac{\Sigma_{i=1}^{N} SIZE_j^i}{max_i(RECV_j^i) - min_i(SEND_j^i)}$. The *completion time interval* is: $T_{recv(j)} = max_i(RECV_j^i) - min_i(RECV_j^i)$ ([8]). The *individual throughput* is: $IT_j^i = \frac{SIZE_j^i}{RECV_j^i - SEND_j^i}$.

7.3. Experiments of Local Testbed

The parameters of local testbed are listed in Table 10 (left). We compare the performance of the basic scheme with the token-ring scheme under two scenarios: (1) streaming as fast as possible and (2) streaming with fixed sending rate.

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

NSG	2
NRG	2
NS	12
fps	4

Figure 10: Parameters of Testbeds

Scenario 1. In this scenario, the controller forwards the token as soon as it gets it. The performance of the basic scheme is shown in Section 5.2. For the token-ring scheme, the performance parameters are plotted in Figure 11.

The average overall thoughput is improved to 75.9 Mbps (basic scheme: 54.1 Mbps) while the average individual throughput is 76.9 Mbps. The individual throughput indicates a less degree of congestion which could be further using *hold after transmit* (HAT) time. That is, after the sender sends out a frame it keeps the token for a short period before forwarding it to the next sender. The experiments with HAT ranging from 1 to 10 ms indicate that the maximum reaches between 1 and 2 ms, showing a 43.9% improvement over the base case (more in [17]).

Scenario 2. In this scenario, the gateway controller applies rate control by keeping the token for certain amount of time before initiating the next round. The experimental results in Figure 12 and 13 compare the overall throughput and receiving interval in both basic and token-ring schemes ($NS = 5$, $FR = 1$ fps, $HAT = 0$).

After 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 video streaming. In the 100 Mbps LAN, it can support up to 6 uncompressed video streams with greater than 4 fps.

7.4. Experiments of Remote Testbed

To our best knowledge, we are the first one to present the results of tele-immersion streaming across 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. The parameters of remote testbeds are listed in Table 10 (right).

For the token-ring scheme, we set $HAT = 1$ ms. We apply the real-time compression scheme for data compression (Section 6.1). The experimental results are listed in Figure 14. The macro-frame delay is the end-to-end delay as: $DELAY_j = max_i(RECV_j^i) - min_i(SEND_j^i)$.

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.

8. Conclusion

We have presented TEEVE, a tele-immersive framework, that integrates the tele-immersive 3D camera array edges with 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 6 uncompressed 3D video streams) across LAN to the service gateways, and (b) we can sustain this 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.

Acknowledgements

We would like to acknowledge the support of this research by the National Science Foundation (NSF SCI 05-49242, NSF CNS 05-20182). The presented views are those of authors and do not represent the position of NSF.

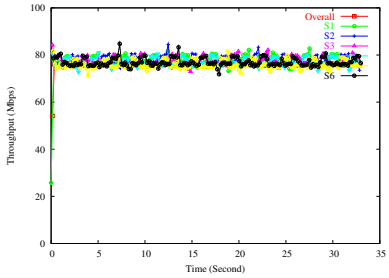


Figure 11: Token-ring Streaming

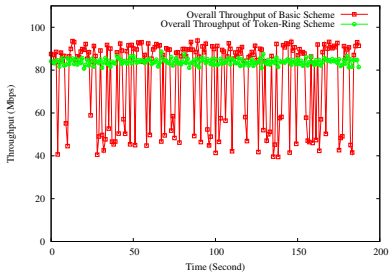


Figure 12: Overall Throughput

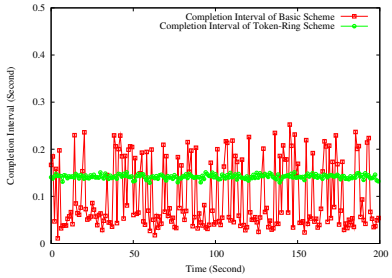
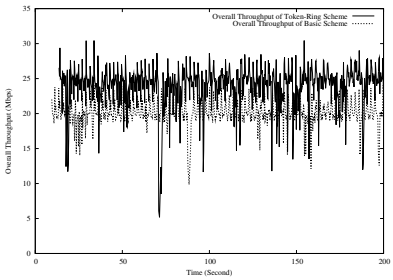
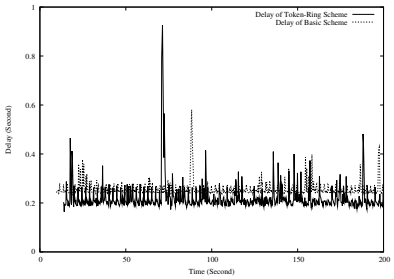


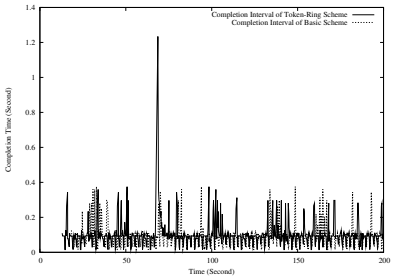
Figure 13: Completion Interval



(a) Overall Throughput



(b) Marco-Frame Delay



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

Figure 14: Streaming Performance of Remote Testbed

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*, 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] 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.

[5] 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.

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

[7] J. Mulligan, V. Isler, and K. Daniilidis. Trinocular stereo: A real-time algorithm and its evaluation. *International Journal of Computer Vision*, 47(1-3):51–61, April 2002.

[8] 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*, pages 596–603, New York, NY, USA, 2004. ACM Press.

[9] 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.

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

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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*, pages 908–915, New York, NY, USA, 2004. ACM Press.

[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. In *Thirteenth Annual Multimedia Computing and Networking (MMCN'06)*, January 2006.

[17] Z. Yang, Y. Cui, B. Yu, J. Liang, K. Nshrstedt, S.-H. Jung, and R. Bajscy. Teeve: The next generation architecture for tele-immersive environments. Technical Report UIUCDCS-R-2005-2631, University of Illinois at Urbana-Champaign, September 2005.

[18] 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.