Enabling Multi-party 3D Tele-immersive Environments with *ViewCast*

ZHENYU YANG, WANMIN WU, KLARA NAHRSTEDT
University of Illinois at Urbana-Champaign
and
GREGORIJ KURILLO, RUZENA BAJCSY
University of California at Berkeley

Three-dimensional tele-immersive (3DTI) environments have great potential to promote collaborative work among geographically distributed users. However, most existing 3DTI systems work with only two sites due to the huge demand of resources and the lack of a simple yet powerful networking model to handle connectivity, scalability, and quality-of-service (QoS) guarantees.

In this paper, we explore the design space from the angle of multi-stream management to enable multi-party 3DTI communication. Multiple correlated 3D video streams are employed to provide a comprehensive representation of the physical scene in each 3DTI environment, and rendered together to establish a common cyberspace among all participating 3DTI environments. The existence of multi-stream correlation provides the unique opportunity for new approaches in QoS provisioning. Previous work mostly concentrated on compression and adaptation techniques on the per stream basis while ignoring the application layer semantics and the coordination required among streams. We propose an innovative and generalized *ViewCast* model to coordinate the multi-stream content dissemination over an overlay network. ViewCast leverages view semantics in 3D free-viewpoint video systems to fill the gap between the high-level user interest and the low-level stream management. In ViewCast, only the view information is specified by the user/application, while the underlying control dynamically performs stream differentiation, selection, coordination and dissemination. We present the details of ViewCast and evaluate it through both simulation and 3DTI sessions among tele-immersive environments residing in different institutes across the Internet2. Our experimental results demonstrate the implementation feasibility and performance enhancement of ViewCast in supporting the multi-party 3DTI collaboration.

Categories and Subject Descriptors: C.2.3 [**Computer-Communication Networks**]: Network Operations; C.2.2 [**Computer-Communication Networks**]: Network Protocols; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design

General Terms: design, performance, experimentation

Additional Key Words and Phrases: 3D tele-immersion, networking protocol, distributed multimedia system, multi-stream coordination, QoS adaptation, application level multicast

1. INTRODUCTION

Three-dimensional tele-immersive (3DTI) environments have great the potential to promote collaborative work among geographically distributed users. Earlier research efforts [Daniilidis et al. 2000; Kauff and Schreer 2002; Raskar et al. 1998] have illustrated the possible applications of

Author's address: Z. Yang, W. Wu and K. Nahrstedt, University of Illinois at Urbana-Champaign, Department of Computer Science, 201 N. Goodwin, Urbana, IL 61801, U.S.A.; eMail: {zyang2, wwu23, klara}@uiuc.edu.
Author's address: G. Kurillo, R. Bajcsy, University of California at Berkeley, Department of Electrical Engineering and Computer Sciences, 253 Cory Hall, Berkeley, CA 94720, U.S.A.; eMail: {gregorij, bajcsy}@eecs.berkeley.edu.
This work was supported by the National Science Foundation (NSF SCI 05-49242 and NSF CNS 05-20182).

3DTI environments in various areas such as scientific research, medical science, artistic performance, education, physiotherapy, training, and entertainment, where a higher level of spatial interactivity is desired. Meanwhile, the end-devices (e.g., 3D cameras and displays) that make the tele-immersive edge applications possible are becoming more available and deployable due to the advances in hardware. Consequently, there have been various efforts to create tele-conferencing and tele-immersive environments [Baker et al. 2005; Chen 2001; Gharai et al. 2002; Ott and Mayer-Patel 2004; Shi et al. 2004; Schreer et al. 2001]. The current approaches represent a very good start for the next generation of tele-immersive systems with an ultimate goal of delivering 3DTI experience to the broader audience.

However, disregarding the promise, extensive deployment of 3DTI environments is still hindered by the problems pertaining to connectivity, scalability and quality-of-service (QoS) guarantees. There are two major deficiencies in the current work. First, most existing 3DTI systems either do not provide 3D multi-stream immersive content or require dedicated computing and networking components. We argue that with the advance of end-devices it is now practical to further extend the application of 3DTI environments with general content creation and delivery infrastructures. Second, most existing systems only support the inter-connection of two parties across the Internet. Enabling multi-party 3DTI collaboration is challenging due to the huge demand in computing and networking resources.

Figure 1 illustrates the multi-party 3DTI environments where remote users meet in a common virtual space and interact three-dimentionally. To accomplish that, the underlying system must capture, process, transmit, and render multiple 3D models with real-time performance. Hence, a critical question is how to organize the large volume of 3D visual data and their corresponding resources, over the current commercially available (COTS) computing and networking infrastructures for the delivery of realistic immersive experience so that "*everybody*" would be able to install and enjoy the 3DTI environments for high quality tele-collaboration.

Fig. 1. Collaborative Multi-party 3DTI Environments

1.1 Characteristics and Challenges

The design of 3DTI environments is based on the *multi-party/multi-stream* application model as illustrated in Figure 2. Each party represents one environment where an array of 3D cameras is installed from various angles to cover a wide field of view of the local scene. Using real-time computer vision techniques, the camera array dynamically derives the 3D model of the user represented in multiple video streams with each corresponding to one camera. The generated and correlated video streams are exchanged with the remote tele-immersive environments through the networking fabric. Finally, the 3D representations from different remote environments are merged

and rendered together, creating a collaborative virtual space with the immersive awareness for every participant.

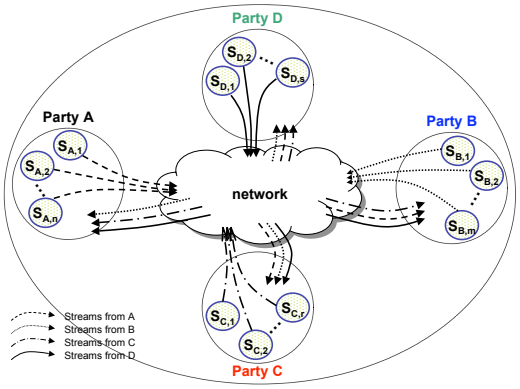


Fig. 2. Multi-party/Multi-stream Application Model

In this paper, we explore the solution of enabling multi-party 3DTI communication from the angle of *multi-stream management* in content delivery. Our work is motivated by addressing the following challenges.

—**Large-volume Data.** To achieve the realistic 3D visual effect, it is desirable to transmit multiple video streams from each 3DTI environment in one communication session. In our experimental system [Yang et al. 2005; TEEVE 2006], one uncompressed 3D video stream has the basic rate over 30 Mbps (at the resolution of 320×240 pixels per 3D image frame and 10 frames per second) to support the spatial collaboration, and each environment produces up to 10 streams. The resolution is being upgraded to 640×480 pixels/frame and over 15 frames/second. If all streams were sent, the overall bandwidth from one environment would soon exceed the Gbps level. The problem becomes even more exacerbated if multiple environments are connected. Suppose the basic data rate of one stream is r, the number of streams per environment is N_s, the number of environments in one 3DTI session is N_e, and the number of 3DTI renderers (i.e., computers that host displays) is N_r. Then the total amount of data to be transmitted will become $O(r \times N_s \times N_e \times N_r)$ which is a very significant demand at the networking scale of Internet.

—**Rendering Cost.** At the current spatial/temporal resolution, one 3DTI environment with 10 streams requires a rendering capacity of 7.7×10^6 points/second. Unlike the 3D reconstruction, the parallelization of rendering is much more difficult as all streams must be rendered in one single virtual space. Thus, the cost of rendering grows linearly with the total amount of stream data sent to the rendering process (i.e., $O(r \times N_s \times N_e)$).

—**Stream Correlation.** In one 3DTI environment, the video streams derived from the 3D camera array are correlated as all cameras are calibrated and synchronized to concurrently capture the visual information of one physical scene. The rendering quality depends on the overall contribution of the streams. Traditional QoS approaches in 2D and single-stream scenario thus become inadequate as the correlation among streams is not considered and hence the coordination of multiple streams is missing. This unique multi-stream content feature, combined with the resource bound imposed by the bandwidth and rendering overhead, demands for the design of a *multi-stream coordination* in the 3DTI content distribution.

—**View-based Rendering.** Unlike 2D video rendering, 3D video rendering is an interactive process. In order to render the 3D models with the correct visual effect, the displaying device needs to keep track of the user view information (using e.g. head-mounted devices or mouse/keyboard) and render the 3D scene accordingly. The interactivity through view selection is the key feature of a 3D video application [3DAV Report 2003]. The problem is how to incorporate the view semantics into the design of a multi-stream management for a more efficient QoS provisioning that is not achievable through the previous 2D/single-stream QoS techniques. Further, we need to point out that the view-based rendering and stream correlation are *dynamic* concepts as the user view could change arbitrarily during one 3DTI session.

—**Multi-party Connectivity.** Finally, the problem of connecting multiple 3DTI environments has become more complicated due to the aforementioned challenges. We emphasize that the scalability model in the 3DTI context is different from traditional peer-to-peer streaming applications. The scalability of the 3DTI systems is not in the number of distributed sites, but in the streaming *density* scale (number of I/O devices in a 3DTI room) among a small number of sites. We do not expect a 3DTI session to consist of more than an order of ten sites. This is because immersed users collaborate jointly in the shared 3D virtual space, the joint display space can become a limitation, and the human's attention space is intrinsically limited. However, the challenge is that a slight increase in the number of sites significantly increases the streaming density of the system due to the multi-stream/multi-site features and the large amount of resources a site consumes. Because of the huge data volume, it is impractical to take the approach of a unicast-based dissemination scheme. However, most available multicast schemes are single-stream-oriented, and do not have the desired flexibility to accommodate the dynamics of stream correlation and view semantics inherent in 3DTI environments.

1.2 Our Contributions

Our solution takes a novel approach by leveraging the unique feature, *view*, in multi-party/multi-stream systems to optimize stream dissemination and coordination, and provide quality guarantees. As mentioned earlier, in 3DTI environments the user view information is required to render a 3D scene correctly. The view defines a particular viewing perspective or viewpoint that the user prefers to observe the virtual space at any given time. Therefore, it is an important concept that would benefit the underlying layers for more efficient content delivery. Unfortunately, traditional networking models (e.g., multicasting) formulated at the streaming content level have no support of any *view*-oriented multi-stream management and content delivery. To bridge the gap, we propose an innovative and generalized *ViewCast* model between the application layer and the underlying delivery layer (e.g., an end-system overlay network). The basic idea is that the user only specifies his/her view interest. The problems including how to dynamically map the user view to an ideal set of streams and how to coordinate multi-streaming are left to ViewCast. The insight is that with an ultimate goal of satisfying the rendering quality of a particular view, the model ensures that the underlying delivery layer can have more flexibility in customizing streaming topology for improved multi-stream coordination and QoS adaptation.

We embed the ViewCast model as a distributed service middleware in our 3DTI implementation of the TEEVE project [TEEVE 2006]. Our collaborative virtual space currently has three major environments (with an average N_s of 8 streams) and a few rendering sites located in different institutions across the Internet. The major contributions of the paper are summarized below.

—**View-oriented Content Dissemination.** We present a novel ViewCast model to coordinate the multi-stream content dissemination on top of the end-system overlay network for supporting multi-party 3DTI communication. Different from all other stream-oriented multicast protocols, the scope of QoS management in ViewCast is not bounded by a fixed set of streams. The new view-oriented approach brings more flexibility, customization and adaptability to the design.

—**Multi-party 3DTI Communication.** To the best of our knowledge, we are the first to present a feasible solution to support multi-party 3DTI communication with a multi-stream 3D video content. Our work will provide valuable reference to the future generation of 3DTI systems that extend towards larger user groups and more interesting collaborative activities.

—**Implementation and Validation.** We implement the prototype of the ViewCast model. The ViewCast-based content distribution allows stable quality view rendering of the common virtual space in a real 3DTI session. Meanwhile, we evaluate the scalability of the ViewCast model through extensive simulation tests to demonstrate the implementation feasibility and potential performance enhancements in efficiency, scalability and quality adaptation under a larger scale of the 3DTI deployment.

The remainder of the paper is organized as follows. In Section 2 we give an overview of the 3DTI environments and the multi-party 3DTI session architecture. In Section 3 we then formalize the maximum quality and minimum quality problem in supporting multi-party 3DTI environments and the ViewCast solution. We present the evaluation results in Section 4. We discuss related work in Section 5, and conclude in Section 6.

2. OVERVIEW

We present an overview of 3DTI environments (more details in [Yang et al. 2005]) and the overlay networking structure (more details in [Yang et al. 2007]) to facilitate further discussion.

2.1 Data Model

The overall 3DTI data model consists of two parts: (a) the *3D reconstructed video data model* that represents the information derived from one single 3D camera, and (b) the *integrated data model* that includes all 3D video streams captured concurrently by the 3D camera array. Figure 3 shows the pictures of the 3D camera and the 3D camera array used in our 3DTI system.

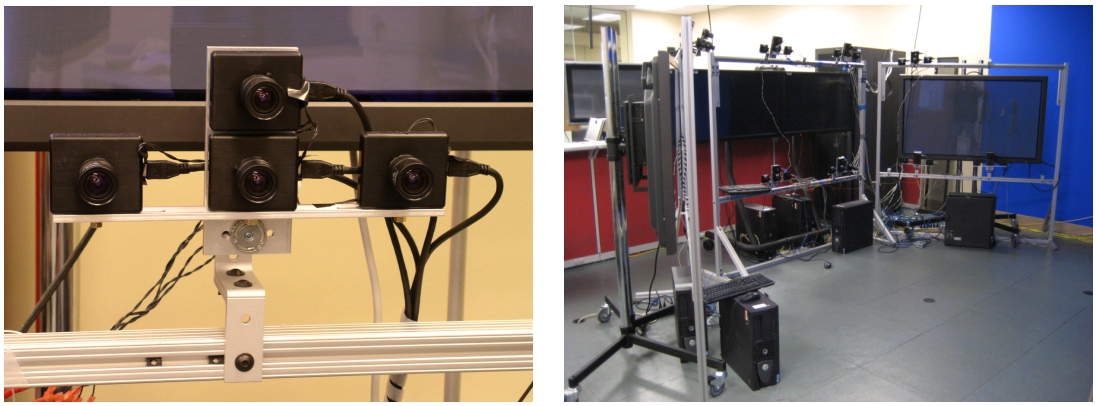


Fig. 3. 3D Camera and Camera Array

3D Reconstructed Video Data Model. One 3D camera consists of multiple 2D digital cameras and their joint processing unit. The 2D cameras are calibrated and synchronized to perform image-based vision algorithm in real time. The output 3D frame contains not only the color but also the spatial depth information per pixel. Thus, each 3D camera produces one continuous stream of 3D frames corresponding to its particular viewpoint in space.

Integrated Data Model. In one 3DTI environment, 3D cameras mounted at various spatial points of a room form a *3D camera array*. All 3D cameras are globally calibrated and synchronized. One dedicated computer is used as the synchronization source, which sends impulse signals

to the external trigger of all cameras via the general purpose input/output (GPIO) pins on the camera board. The average time from when the trigger register is written to the start of the integration of the image frame is approximately 56 μs, which allows for very high synchronization accuracy (ignoring the signal propagation delay). Therefore, the set of 3D frames bearing the same timestamp, denoted as one *macro-frame*, constitutes a comprehensive panorama-like representation of the physical scene. Note that, except for calibration and synchronization, each 3D camera is an independent unit in terms of the visual content.[1]

2.2 View Model

One of the unique characteristics of the 3D camera array is its *stream correlation* in the sense that the 3D cameras are concurrently capturing data with a synchronized clock and presenting a complementary visual information of a common physical scene. However, since a user only observes the visual information from one particular view at any given time and each 3D stream only conveys part of the whole content from its own viewpoint, the *contribution* of each stream to the current user view must be different. For example, if the user is currently looking at the front of a 3D object, streams generated from side cameras are not as important as front cameras (regarding to the object). If we assume the object is not transparent, which is usually valid in the 3DTI case, then the cameras from the back are the least important. Such view-dependent stream differentiation can be formally represented using the view model.

More specifically, we are interested in deriving an appropriate indicator of the stream importance regarding to the user view. One example of the stream differentiation function is given in [Yang et al. 2006]. Basically, let us denote $s.\vec{w}$ as the unit vector representing the spatial orientation of the stream s and $u.\vec{w}$ as the unit vector representing the view of user u. Then the differentiation function (denoted as df) calculates the stream importance as in Equation 1.

$$df(s, u.\vec{w}) = s.\vec{w} \cdot u.\vec{w} \tag{1}$$

In the above equation, the value of the dot product is $cos\theta$, where θ is the angle between the vectors of $s.\vec{w}$ and $u.\vec{w}$. The calculation is based on an important observation that when a camera moves away from the viewing direction of the user, its effective image resolution as projected onto the viewing plane will decrease due to foreshortening and occlusion. When $s.\vec{w}$ and $u.\vec{w}$ are close to each other, the dot product is close to 1, showing that the stream s is very important to the user view. Otherwise, the value decreases to -1 indicating less importance. The effect is illustrated in Figure 4. Given the user view (the right part), cameras 4 and 5 are the most important ones. Cameras 3 and 6 are less important but will improve the visual quality if added. The rest of the cameras are the least important. As shown in [Yang et al. 2006], the quality adaptation can be achieved by simply dropping the streams whose importance is below a certain differentiating threshold.

2.3 Multi-party 3DTI Session Architecture

Figure 5 illustrates the 3DTI session architecture which is based on an end-system overlay network with two hierarchical levels. At the *local* level, each 3DTI environment is managed by its *service gateway* (SG) which consists of one or more dedicated processors. When a 3D camera is initiated, it registers with its service gateway to save the meta-data. Due to the overhead of the 3D reconstruction, once a 3D frame is generated it is forwarded to the service gateway through the high-speed LAN for the further processing of data compression and streaming control. The service gateway helps with the rendering as well by retrieving streams on behalf of its local renderers.

[1] Due to the one-to-one correspondence between a 3D camera and its generated 3D video stream, we will use these two terms interchangeably.

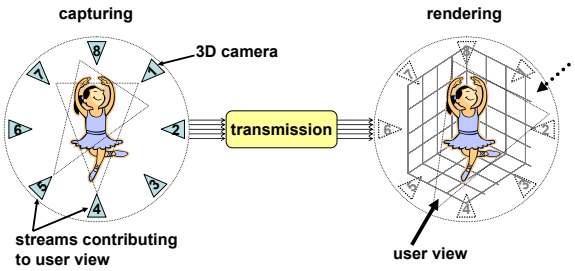


Fig. 4. Stream Differentiation regarding to *View*

Thus, the service gateway represents an application level data aggregating point at each 3DTI environment.

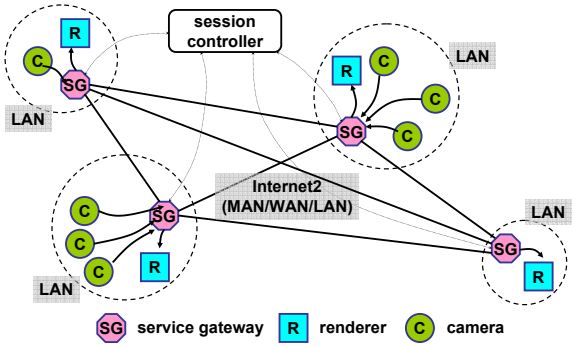


Fig. 5. The Overlay Network of a Multi-party 3DTI Session

After the bootstrapping of the local environment is completed, the service gateway registers with the central *session controller* at the *global* level. The way the session controller organizes the multi-party 3DTI communication is similar to other proposed schemes (e.g., [Hosseini and Georganas 2003]). When a new service gateway joins the session, the session controller informs other service gateways to let them interconnect and form the initial graph for content delivery. For simplicity, it is assumed that all participating service gateways must register with the session controller before the live session can start. We take a centralized approach at the global level because of its low messaging cost and responsiveness to the dynamics of each 3DTI session. The approach is feasible in our situation since the number of service gateways is within a reasonable scale (≤ 10).

We design the ViewCast model based on the application level multicast, which is becoming an appealing alternative to IP multicast due to the advantage of flexibility, easy deployment and low overhead [Chu et al. 2000]. At the application level, the service gateways collect multiple streams either from the local sources (i.e., cameras) or from the remote sources (i.e., peer service gateways). The collected streams are then multicasted according to the current status of the overlay network, the view change requests and the forwarding schedule managed by the session controller.

Figure 6 illustrates how ViewCast works. Each vertex (i.e., v_1, v_2, v_3, v_4, v_5) represents a service gateway. The service gateways form an overlay network and cooperate for content delivery. A vertex can request a view from a multi-stream source on behalf of its renderer. More specifically, the renderer provides users an interface for manipulating the viewpoint (e.g., by using mouse).

When a view change is detected, the new view is sent to the local service gateway (refer to Figure 5). For example, the renderer u_2 registers with the vertex v_2 and v_2 requests a view (denoted as $u_2.w$) from vertex v_5. Depending on the view request and available resources, each requesting vertex may get different subset of streams. As long as the quality and resource constraints are satisfied, the vertices that have available streams can serve other vertices. Furthermore, a vertex can retrieve streams from multiple vertices in parallel.

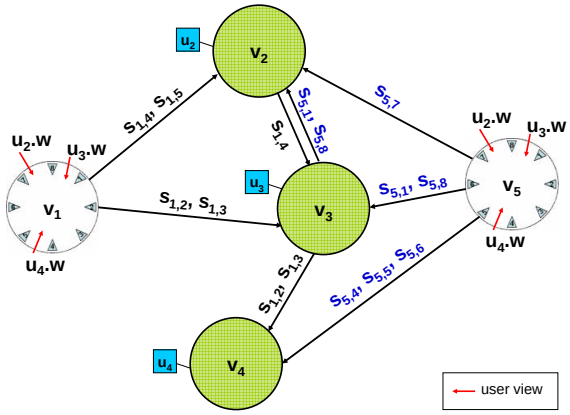


Fig. 6. ViewCast-based Multi-streaming

During a live 3DTI session, the user can switch his/her viewing position of the virtual space at the renderer. If the view change cannot be accommodated due to the lack of required streams, the renderer will forward the request to its local service gateway. The service gateway checks whether it has the streams available for serving the view change. If not, it sends a view request to the session controller to compute a new multicast topology for coordinating the multi-streaming. After a new dissemination topology is calculated, the session controller broadcasts it to all service gateways to complete the view request. Since a macro-frame may travel along different overlay paths, buffering and timeout mechanisms are used at the service gateways and the renderers to synchronize the transmission. Synchronization and delay management are very critical issues under such scenario and will be further investigated.

3. VIEWCAST

The view concept reflects the user interest at a higher level, which distinguishes the ViewCast from any other content delivery schemes at the per stream level. The ViewCast scheme basically specifies that when the user retrieves content from a multi-party/multi-stream system as in the case of the 3DTI environments, only the user's view interest is required. The ViewCast scheme then controls the stream selection dynamically according to the view requirement and the status of resources with the ultimate goal of sustaining the QoS for the rendered view. Therefore, ViewCast has the advantages of improved flexibility, customization, adaptability, coordination and responsiveness under more dynamic and resource-constrained environments. We envision the application of ViewCast in, for example, multi-camera conferencing, surveillance system, 3D TV, and video sensor networks. We believe that delivering an experience enriched with a plethora of media devices, contents, and services will become the theme of research for the next generation distributed multimedia applications. Those applications are characterized by a complex merging of: (a) multiple homogeneous and/or heterogeneous media streams that are semantically correlated, (b) significant resource and attention constraints, and (c) application and user

oriented requirements of configuration, adaptation, composition, coordination and synchronization. The concept of integrating user interest into a smart, automatic and responsive system management will play a very important role. From the 3DTI prospect, we point out several important properties of ViewCast as listed below. Most of the features may apply to other multi-party/multi-stream systems as well.

—**Stream Correlation.** The basic assumption for ViewCast is that each content source supplies multiple correlated streams. Stream correlation is an important feature in multi-camera systems, where the concept of view has a very intuitive definition.

—**Stream Differentiation.** Along with the stream correlation is the feature of stream differentiation. That is, a given view should favor some of the streams over others as described in Section 2.

—**Inter-stream Coding Independency.** We assume the coding/decoding independency among streams. Since each stream can be independently transmitted and rendered, it is easier to perform the "view to stream" mapping and to select streams with different possible combinations. As illustrated in Figure 4, given the user view and the orientation of cameras, ViewCast can select various subsets of streams (i.e., cameras) such as $\{s_4\}$, $\{s_4, s_5\}$, $\{s_4, s_5, s_3\}$ or $\{s_4, s_5, s_3, s_6\}$... depending on the quality and resource constraints. Inter-stream coding independency provides more flexibility in the stream selection and its QoS adaptation. On the other hand, it also adds design challenges as there are more choices.

—**Open Model.** From an OSI layering point of view, the ViewCast scheme resides in the presentation layer. It maps between the semantics in the application layer (i.e., stream correlation/differentiation) and the streaming manipulation in the session layer. More specifically, in 3DTI environments the ViewCast scheme only dictates the desirable attributes of the application (e.g., the definition of view) and how these attributes affect the multi-streaming. However, the design choice of the higher and lower layers is open depending on the specific requirements. We are the first to identify ViewCast as a very core and ideal function in the presentation layer.

—**View Change.** As observed, the view change operation may occur frequently in 3DTI environments. There are two consequences of the view change. First, stream differentiation varies with a view change. For example, in Figure 4 when the user view changes to the position of dotted arrow (right part) cameras 1 and 2 will become the most important ones. This variance distinguishes ViewCast from other systems that only consider the fixed stream differentiation such as the layered coding and the multiple description coding. Second, as soon as the view change is detected, the system must respond by switching streams accordingly. Stream switching may be costly. The direct impact to the user may be the discontinuity or delay of view rendering. If the multicast protocol is used in the underlying layer, then the stream switching at the parent vertex may influence child vertices. The dynamics of view change presents a critical challenge for designing the ViewCast model.

The key insight for ViewCast is that by leveraging the high-level view semantics the visual quality, which is closely related to view, can be improved in a more orchestrated way by allowing the low-level streaming regulation layer larger flexibility for delivery coordination and QoS adaptation so that the resource constraints can be satisfied adaptively.

3.1 Problem

With the diverse user view interest, the most critical issue of ViewCast is how to organize the content distribution topology such that it is more efficient for the balance of resource and quality. For example, it is desirable to guarantee that the most important streams for all view requests could always be delivered (i.e., *Minimum Quality Problem*) or to maximize the set of important

streams for all view requests (i.e., *Maximum Quality Problem*). However, those problems are generally NP-hard under various system constraints including the bandwidth constraints, the relay constraints, and the delay constraints (more details in [Yang et al. 2007]). If we further consider the dynamics of the user view interests, those problems will become more complicated.

Note that, an ideal criterion of stream importance should be based on the rendering quality. However, in 3DTI environments the rendering quality of particular view depends on the set of streams received. Since it is quite complicated to derive an exact form of the quality function with a given set of streams, a simple linear approach is taken to evaluate the *relevance* of a set streams S' to a view w as shown in Equation (2),

$$relv(S', w) = \sum_{s \in S'} df(s, w) \tag{2}$$

where *df* denotes the *differentiation function* which gives the relevance of a stream regarding to a given user view. Depending on a specific application, the definition of the differentiation function could be different. In the case of 3DTI environments, the differentiation function is given in Equation 1. The above function of relevance is what we use as the basis to formulate and investigate the problems as we mention earlier. For example, the minimum quality problem can be formulated as below.

Minimum Quality Problem

1. to satisfy: $\forall v_i \in V, relv(R_i, v_i.w) \geq \Delta$
2. subject to bandwidth, delay, and system availability constraints

where Δ is a given lowest bound, R_i represents the set of streams that node v_i is currently receiving, and $v_i.w$ represents the view of v_i. Although in the rest of the discussion we keep using the term *quality*, it should be clear that the quality of a set of streams refers to the overall relevance instead of the final rendering quality.

3.2 Solution

We have two major designing goals for the solution of ViewCast.

—**Minimum quality guarantee.** Each vertex should receive a minimum set of streams to have some rendering quality guarantee of every other vertex inside its view. For 3DTI environments, it implies the consistent presence of all participants in the virtual space, which is critical for the collaborative work.

—**View change resilience.** When a vertex changes its view, the impact on other affected vertices should be minimized for the continuity of group interaction.

3.2.1 Minimum Quality Guarantee

Because the minimum quality problem is NP-hard, we propose heuristics using the approach based on priority [Yang et al. 2006] and preemption [Yang et al. 2007] with the following steps.

Step 1. Given a view request $v.w$, the importance of the stream is calculated using the differentiation function $df(s, v.w)$.

Step 2. The streams are selected if the *df* value is above a certain threshold. For example, in our 3DTI environment we choose the streams with $df(s, v.w) \geq 0$, reflecting a 180° total view range.

Step 3. The selected streams are further differentiated into several priority groups according to their importance. We define the set of priorities P as $\{p_1, p_2, ..., p_k\}$, where p is in ascending order, i.e., $p_i < p_{i+1}$ for $i = 1$ to $k - 1$. We assign priorities to the selected streams according to the differentiation function. That is, we sort the selected streams according to the value of the differentiation function. The stream with the largest value of the differentiation function is assigned the highest priority p_k, the stream with the second largest value is assigned the the

priority p_{k-1}, and so forth. The result is that each priority group (with priority p_i) typically contains one stream from each site. The choice of the parameter k depends on the number of streams originating from each vertex. In 3DTI environments each vertex has around 8 streams. The stream selection in Step 2 produces a subset of 3 to 4 streams. Therefore, we set k to be 4.
Step 4. As mentioned earlier, the inbound (and outbound) bandwidth resource is divided into bins with each bin hosting one stream. Suppose it is needed to forward a stream s from the vertex v_i to the vertex v_j. If both vertices have available bins, it is straightforward to establish the streaming. Otherwise, the bin of lower priority stream can be preempted. For example, if the stream s has p_4 priority based on the view, it can take the outbound (or inbound) bin occupied by the stream of a lower priority (i.e., p_1, p_2, p_3) in either v_i or v_j respectively. When the preemption is needed, the bin of the lowest priority stream will be taken first. The bin allocation of the selected streams is performed in the descending order of priority and terminated when the preemption is not possible.[2] Note that the broken links will be repaired by a `fix_victim()` routine described in the next section.

The preemption mechanism is important in the resource-constrained 3DTI environments. Since bandwidth is such a sparse resource, prioritization-based preemption is a key to guarantee *efficient* utilization of the resource in times of contention. Further, preemption is not "intrusive" (in the sense of visual quality disruption) due to the 3D data representation. All streams are aggregated and rendered into a single 3D representation of human body, which is very different from the conventional 2D video systems where losing a stream means losing a scene (e.g., the window showing that stream). Here when the user is observing the scene from the front view, for example, it will not be as visually noticeable if some stream from the side view is preempted. The 3D pixel cloud of the participant is still being rendered, only that some of the less important points (in terms of view) are absent. The released resource can then be used to serve higher-priority streams which contribute more to the overall visual quality.

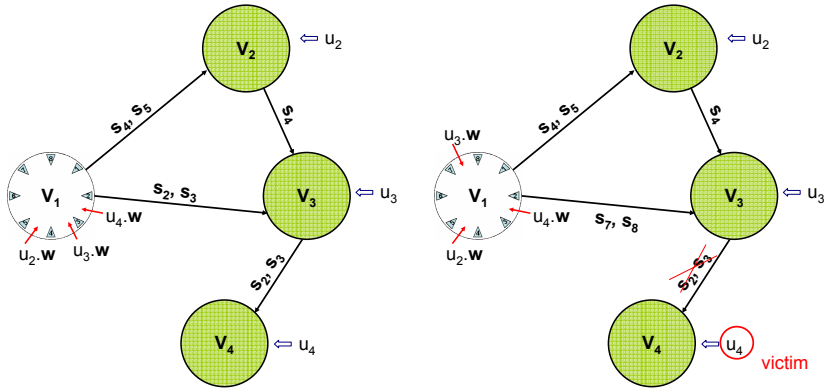
3.2.2 View Change Resilience

The negative impact of the view change is illustrated in Figure 7. In the figure, the vertex v_3 and v_4 have similar views and v_4 is streaming s_2 and s_3 from v_3. When the vertex v_3 changes its view, streams needed by v_4 may temporarily become unavailable. In such a case, v_4 has *victims* (i.e., broken streams). The impact will grow as the number of dependent vertices increases. The view change operation is a frequent phenomenon in 3DTI environment and may cause large overhead if not treated properly. Note that, the streaming preemption as introduced in Section 3.2.1 may also cause an involuntary termination of streaming in a way similar to the view change.

In a related domain, there are previous solutions dealing with a multicast member join/leave which rely on the concepts of soft leave [Hosseini and Georganas 2003], buffering [Cui and Nahrstedt 2003] and rearrangement [Bauer and Varma 1997; Sriram et al. 1999]. Soft leave requires the changing vertex to continuously serve old streams until affected vertices have found replacement. Although doable, under multi-stream scenario it would incur longer delay. Buffering let the intermediate vertices continue streaming from cache to absorb the propagation of quality degradation. However, it is not a feasible approach for live communication. Rearrangement involves large overhead which requires the multicast content delivery schedule to be relatively stable to amortize the cost.

As we observe, none of these solutions would directly fit in the case of 3DTI environments. Instead, we apply the strategy of *dependency balancing* to improve the resilience of view change tolerance. There are three basic techniques in dependency balancing: (1) *source balancing*, (2)

[2] Currently, ViewCast is operated at the per stream granularity. When there is not enough resource to transmit a stream at its full content, the streaming will be dropped. However, it is an interesting problem to explore whether a quantitative improvement could be achieved at finer granularity (e.g., to transmit a stream with different rates).



(a) Before View Change (b) After View Change

Fig. 7. Effect of View Change

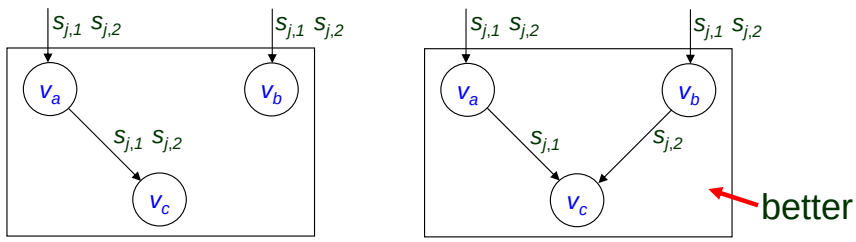


Fig. 8. Source Balancing in ViewCast

priority balancing, and (3) *forwarding load balancing*.

Source Balancing. Source balancing attempts to diversify the supplying vertices to lower the dependency on each individual vertex. The basic idea is illustrated in Figure 8, where vertices v_a, v_b have the same set of streams (i.e., $\{s_{j,1}, s_{j,2}\}$) that can be relayed to vertex v_c. Under that, the streaming schedule on the right part is considered better since it provides a more even distribution of streaming among the sources.

Priority Balancing. The basic idea of priority balancing is illustrated in Figure 9. In the figure, vertices v_a, v_b have the same set of streams (i.e., $\{s_{j,1}, s_{j,2}, s_{j,3}, s_{j,4}\}$) that can be relayed to the vertex v_c. The stream priority of v_c as determined by its view differentiation function is $s_{j,1} : p_4$, $s_{j,2} : p_3$, $s_{j,3} : p_2$ and $s_{j,4} : p_1$. Under that, the streaming schedule on the right part is considered better since it provides a more even distribution of streaming quality among the sources.

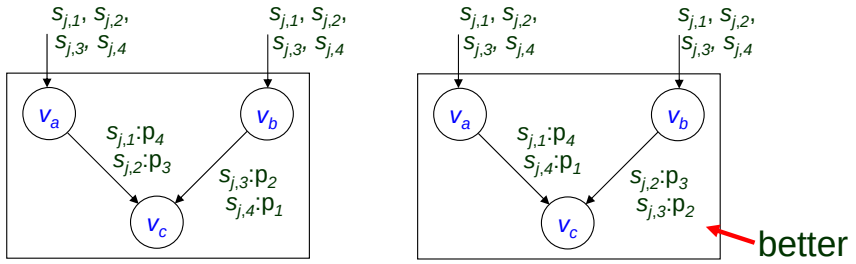


Fig. 9. Priority Balancing in ViewCast

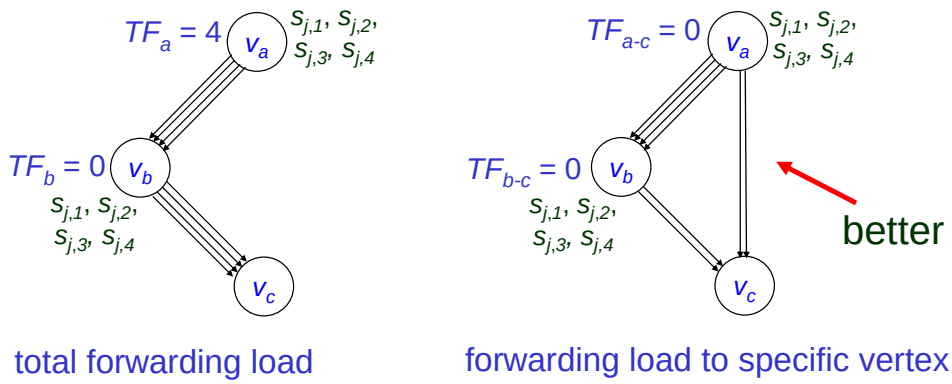


Fig. 10. Forwarding Load Balancing in ViewCast

Forwarding Load Balancing. Forwarding load balancing attempts to balance the forwarding load among all the vertices. There are two criteria in calculating the forwarding load. One is the *total forwarding load* (TF_v)as is often discussed in the literature (e.g., [Castro et al. 2003]). The other is the *forwarding load to a specific vertex* ($TF_{v-v'}$). The ViewCast scheme considers the latter as more important. The justification is illustrated in Figure 10. In the figure, the vertex v_b receives four streams from the source v_a. Next, v_c requests similar streams. Considering the total forwarding load, v_b will be the forwarding vertex to v_c, because before v_c joining, $TF_a = 4$ and $TF_b = 0$. However, that makes v_c solely depend on v_b. In the right figure, if we use the forwarding load to specific vertex as the criterion, then both v_a and v_b can serve v_c because $TF_{a-c} = TF_{b-c} = 0$ (before v_c joining), which reduces the forwarding dependency among the sources. In ViewCast we choose the forwarding load to specific vertex as the first criterion over the total forwarding load due to the dynamics of the view change.

3.3 ViewCast Management

The main task of the ViewCast management is to serve the view request $v_i.w$, which is performed by the session controller for each vertex v_i. The main `serve_view` algorithm is sketched in Table I.

The `get_streams` routine calculates the differentiation function $df(s, v_i.w)$ with the given view $(v_i.w)$. The selected streams are assigned priority and saved in $\bar{S}$.

The `find_source` routine searches for a supplying vertex that can stream s to the vertex v_i while obeying system constraints. It first scans the vertices which have available bins. Then it picks up the vertex that has the minimum forwarding load to the vertex v_i for load and source balancing (break tie with the minimum total forwarding load). If such vertex is not available, it looks for a vertex which has the bins that can be preempted. For priority balancing, each vertex maintains the sum of priorities (sp) for every other vertex. For example, if the vertex v_i serves p_3 and p_2 streams for vertex v_j, then $sp_i(v_j)$ will be 5. The bigger this sum the higher the streaming quality that vertex v_i serves vertex v_j. Therefore, when there are several candidate vertices for relaying one stream to a destination vertex, the one with the smaller sum will be selected to achieve priority balancing.

Recall that the inbound and outbound bandwidth resource is divided into bins with each bin hosting one stream. The `find_out_bin` and `find_in_bin` routines are pretty straightforward. They return either an unused bin or a bin used by a lower priority stream for preemption. More specifically, `find_out_bin` checks the candidate vertex (v) for its outbound bin availability, and `find_in_bin` examines the requesting vertex (v_i) for its inbound bin availability. If there is an available bin which is not used for any stream, it is returned to the caller. Otherwise, the two

Table I. View Management Algorithm

```
serve_view(v_i, v_j)  /* vertex v_i requesting view of v_j */
    S̄ ← get_streams(S_j, v_i.w)
    for s ∈ S̄ in descending order of priority
        v ← find_source(s, v_i)
        if (v found)
            out ← find_out_bin(s, v, v_i)
            in ← find_in_bin(s, v, v_i)
            if (out found and in found)
                serve_stream(s, v, out, v_i, in)
            end
        end
    end
    fix_victim()
end
find_source(s, dst)
    V_1 ← ∅
    V_2 ← ∅
    for each v_k ∈ V other than dst
        if (v_k can stream s to dst under system constraints)
            if (|F_k| < |O_k|)
                V_1 ← V_1 ∪ v_k
            else
                V_2 ← V_2 ∪ v_k
            end
        end
    end
    if (V_1 ≠ ∅)
        select v ∈ V_1 with dependency balancing
        return v
    else if (V_2 ≠ ∅)
        select v ∈ V_2 where v has preemptable outbound bins
        return v
    end
    return not found
end
```

routines look for lower priority streams for bin preemption. For selecting an outbound bin to be preempted, we prefer to choose the lowest priority stream. For selecting an inbound bin to be preempted, one important consideration is to select a stream that is least used in forwarding to reduce the preemption cost, because once an inbound stream is preempted all child vertices that rely on it will not be able to stream from the parent vertex any more.

The `serve_stream` maintains the bookkeeping of inbound and outbound bins of source and destination vertices. When preemption or view change is performed, the affected vertices are saved in the victim set. The `fix_victim` routine tries to fix the broken links. More specifically, it generates a new request from the victim vertex for the lost stream, and tries to process the request using the `find_source` routine. To reduce the cost, this routine only fixes the broken link related to higher priority streams (e.g., p_4 and p_3). For low-priority link, the affected vertex will simply ignore the lost stream and propagate the message to its child vertices. The propagation will terminate either when the preempted stream is not important for all child vertices or some child vertices have found new sources.

After the `serve_view` routine is completed, the session controller calculates the new content delivery graph and broadcasts to all vertices (i.e., service gateways).

4. EVALUATION

For evaluation, we have conducted both simulation and Internet experiments. The simulation experiments help us to analyze the scaling property of ViewCast with larger session sizes. Meanwhile, the Internet experiments provide us with a more practical sense of performance under the real environment with real users. In the rest of this section, we present the schemes, metrics, and results of our simulation and Internet experiments.

4.1 Simulation Evaluation

For a more extensive evaluation, we simulate the ViewCast model under the multi-party 3DTI scenario with a message-driven simulator written in C++ (on Windows and Linux operating systems). More details of the simulation setup are provided in the following sections.

4.1.1 Experiment Setup

The simulator first generates the graph of an application-level overlay network. We use the mesh topology for the overlay network, where the connectivity between vertices follows the uniform distribution. The total number of vertices, defined as the *session size*, ranges from 5 to 10. Since the human attention space is intrinsically limited, we do not expect the size of a session to be larger than ten sites. The total number of edges is determined by *connectivity ratio* (CR) which is the ratio of edges compared with the corresponding complete directed graph. For example, a directed graph of 8 vertices will have 42 edges, if the connectivity ratio is set to 75% (i.e., $56 \times 75\%$). We choose the connectivity ratio from 25% to 100%.

In the experiment, each vertex has 8 original streams which are evenly distributed in the 360° space. For any view request to a vertex, at most 4 of its original streams are selected for an optimal coverage of 180°. To determine the range of inward/outward degree bound, in the maximal case of 10 vertices each vertex requires at least an in-degree of 36 to get the optimal coverage from every other vertex and an out-degree of 8 so that all its streams may be accessible for serving any view. The estimation has not considered relaying overhead. For simplicity, we set the in-degree bound to be the same as the out-degree bound (later we evaluate the heterogeneity in the Internet experiments). Recall that the in-degree and out-degree of a vertex are in the number of streams (Section 3). We set the *degree bound* (DegB) to be 12 (to simulate a small-capacity network) or 36 (large-capacity network) respectively. These numbers are chosen according to our Internet measurements.

During the simulated 3DTI session, each vertex sends the view change request to the central session controller. The view change interval follows the normal distribution with a mean of 60 seconds. The total running time of one simulation experiment is 200 minutes. We use two patterns of the view change: random walk and Zipf. In the random walk pattern, each vertex adds a view change degree to its current view. The view change degree follows the normal distribution with a standard deviation of 20°. We choose the Zipf distribution for the pattern of view change because it is recognized as the common selection pattern for multimedia data [Sripanidkulchai et al. 2004]. The view is changed according to a Zipf distribution of 10 pre-selected view degrees (i.e., $n = 10$). The Zipf distribution is actually the power-law distribution in discrete form. In our 3DTI simulation, it dictates that the ith most popular view degree has the access frequency in proportion to ω/i^α, where α is a constant (we choose $\alpha = 1.0$) and ω is determined by n.

The propagation delay along each edge follows the normal distribution with a mean of 50 ms. We assume each stream has the same bandwidth. The simulation parameters are summarized in Table II.

4.1.2 Rejection Ratio

We measure the *rejection ratio* of each view change request. Recall that in ViewCast, the view change request is served with relevant streams in descending order of importance and the request

Table II. Simulation Parameters

overlay topology	mesh topology
number of vertices	5, 6, ..., 10
connectivity ratio (CR)	25%, 50%, 75%, 100%
in-degree (out-degree) bound (DegB)	12, 36
number of streams per vertex	8
number of streams for optimal view coverage	4
simulation time	200 minutes
view change interval	normal distribution ($\mu = 60$ seconds)
view change pattern	Random Walk (Rwk) and Zipf ($n = 10$, $\alpha = 1.0$)
propogation delay of each edge	normal distribution ($\mu = 50$ ms)

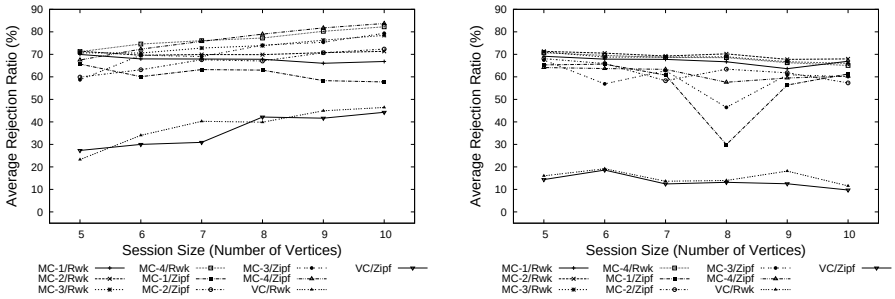
is *rejected* if and only if there is no resource available to supply any stream. For comparison, we introduce another method of the view dissemination based on multicast, where a view request is served by multicasting its relevant streams. However, the application needs to explicitly specify the set of streams needed for a view, and the system must supply all specified streams or reject the request in case of resource limitation. We use the notation of MC-n ($1 \leq n \leq 4$) to refer to the multicast-based method where the application always specifies a set of n streams to satisfy a view request. For example, MC-3 means that the system must deliver 3 streams to the application otherwise the view request will be rejected. Therefore, the MC-n methods differ from ViewCast in the admission control policy. The overlay construction algorithm is the same. Furthermore, MC-n does not apply preemption which implies that the bandwidth resource and the streaming are only released by the receiver. Therefore, in such case it will not cause any victim. But the price of no preemption (i.e., no victim) is the so-called priority inversion problem. The low-priority streams (in terms of semantic importance) can block the high-priority streams from being delivered. ViewCast ensures that the bandwidth is efficiently and appropriately untilized by preemption and fixing victims promptly when needed. The results of the rejection ratio are plotted in Figure 11. We use *VC* to denote the ViewCast method. Other notations are explained in Section 4.1.1.

From Figure 11, we observe that the connectivity ratio has a very strong impact on the view dissemination capacity of the overlay network. In the most constrained cases where CR = 25%, the rejection ratio is pretty high in either ViewCast or the multicast-based methods. However, when the session size increases (≥ 8) ViewCast starts to perform much better. There is a clear-cut in performance at the medium region of CR = 50%. When both the connectivity ratio and the degree bound are improved (CR = 75%, DegB = 36), the rejection ratio of ViewCast becomes almost zero. The rejection ratio of multicast-based method drops as well. However, we show later that even in those cases ViewCast still performs better than MC-n.

4.1.3 Streams Per View

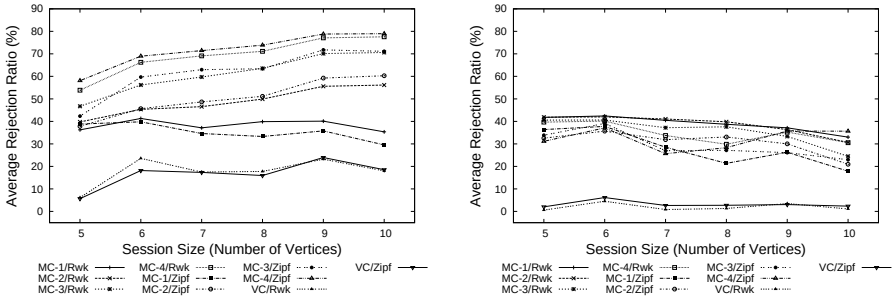
We measure the average number of streams for all successfully served view requests in ViewCast. The results of streams per view are plotted in Figure 12.[3]

Those figures indicate that ViewCast serves each view request with an average of 2 to 3 streams. This number is close to what could be achieved by MC-2 and MC-3. However, if we examine Figure 12 along with Figure 11 and compare ViewCast with MC-2 and MC-3, we will see that ViewCast achieves a much lower rejection ratio by a margin around 20% to 40% in most cases.



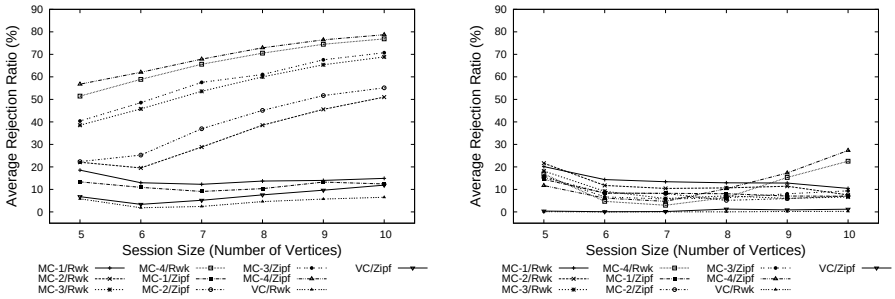
(a) CR = 25%, DegB = 12

(b) CR = 25%, DegB = 36



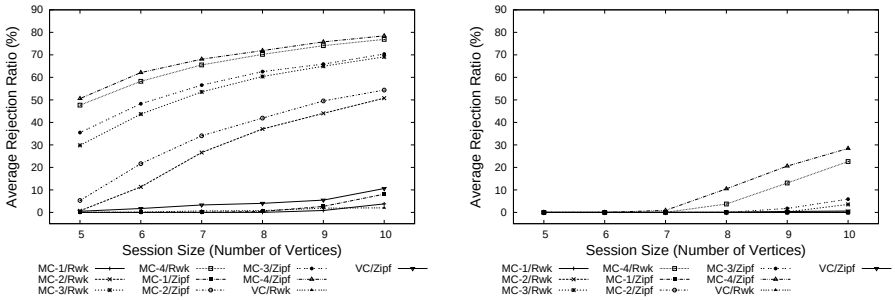
(c) CR = 50%, DegB = 12

(d) CR = 50%, DegB = 36



(e) CR = 75%, DegB = 12

(f) CR = 75%, DegB = 36



(g) CR = 100%, DegB = 12

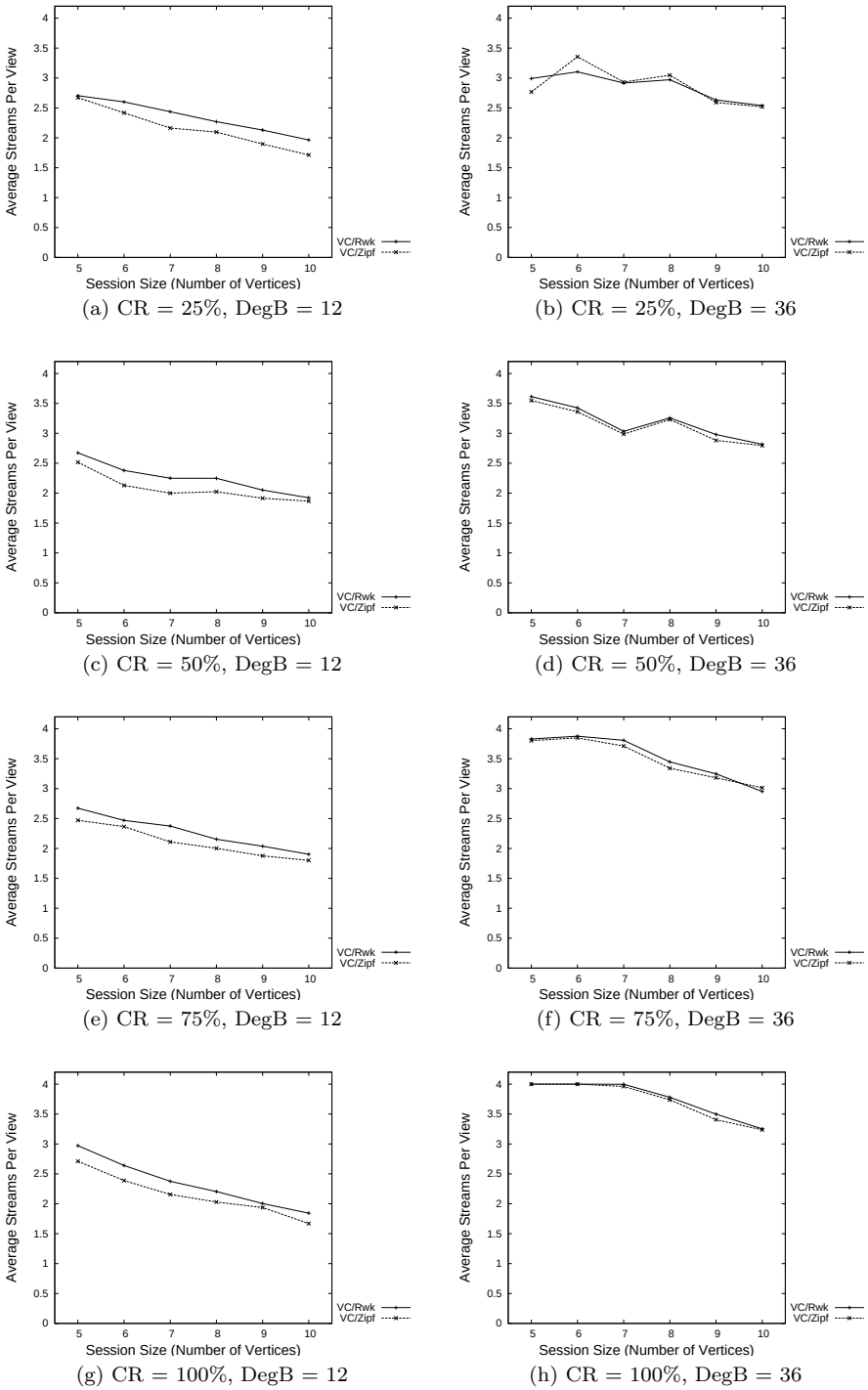
(h) CR = 100%, DegB = 36

Fig. 11. Average Rejection Ratio

4.1.4 Workload

We measure the standard deviation of workload among vertices to investigate how evenly the forwarding load is divided. The workload is defined as the number of out-bound streams served

[3] There is no need to plot the results of streams per view for MC-n as the number will be fixed by the constant of n.

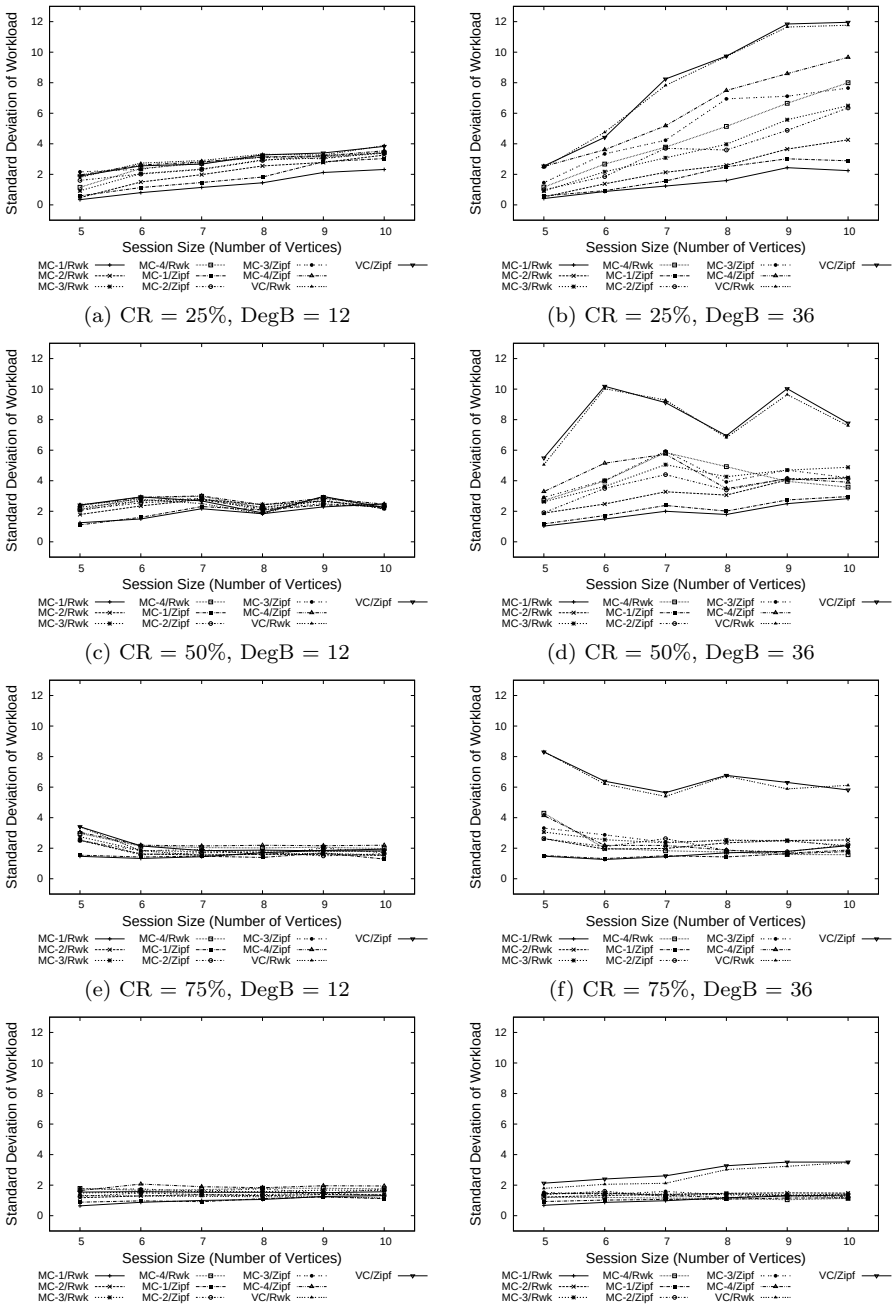


(a) CR = 25%, DegB = 12
(b) CR = 25%, DegB = 36
(c) CR = 50%, DegB = 12
(d) CR = 50%, DegB = 36
(e) CR = 75%, DegB = 12
(f) CR = 75%, DegB = 36
(g) CR = 100%, DegB = 12
(h) CR = 100%, DegB = 36

Fig. 12. Average Number of Streams Per View

by a vertex. The results of the workload standard deviation are plotted in Figure 13.

Those figures indicate several trends. First, the standard deviation grows bigger as the degree



(a) CR = 25%, DegB = 12

(b) CR = 25%, DegB = 36

(c) CR = 50%, DegB = 12

(d) CR = 50%, DegB = 36

(e) CR = 75%, DegB = 12

(f) CR = 75%, DegB = 36

(g) CR = 100%, DegB = 12

(h) CR = 100%, DegB = 36

Fig. 13. Standard Deviation of Workload

bound increases. Second, when the overall demand in stream sharing increases (i.e., MC-4 or MC-3) the workload tends to be more deviated. Third, we notice in terms of load sharing ViewCast tends to perform not as good as multicast-based schemes, which is expected as ViewCast prefers forwarding load to a specific vertex as the major load balancing criterion.

Table III. Bandwidth Measurement

	UCB	UIUC	NCSA	UCF
UCB →		41.8 Mbps	41.6 Mbps	39.3 Mbps
UIUC →	33.2 Mbps		538.7 Mbps	36.8 Mbps
NCSA →	33.4 Mbps	162.1 Mbps		38.1 Mbps
UCF →	36.4 Mbps	49.9 Mbps	38.9 Mbps	

Table IV. End-to-End Application Delay Measurement

UCB ↔ UIUC	32 ms	UIUC ↔ NCSA	0.41 ms
UCB ↔ NCSA	32 ms	UIUC ↔ UCF	25 ms
UCB ↔ UCF	53 ms	NCSA ↔ UCF	26 ms

4.1.5 Collateral Cost of ViewCast

In ViewCast, due to the preemption of the network resource and the view change some vertices may have their streams involuntarily discontinued. We call those broken links the *victims*. More precisely, a victim is defined as a pair of (v, s) where $v \in V$ and $s \in S$. After each view request is served, the routine of `fix_victim()` is called to fix any possible victims (refer to Table I). In the simulation, we measure the average number of victims. Inside `fix_victim()`, if the number of victims is greater than zero then we include it in the computation of the average number of victims. In other words, there are times the number of victims is 0 (in Internet evaluation, we find this to be the majority which is about 88%). The size of the victim set indicates the scope of affected vertex/stream pairs and the expected cost of fixing them. The results are plotted in Figure 14.[4]

The results indicate that in most cases the number of victims is small (around $2 \sim 3$). Note that these broken links will be fixed by using a new parent vertex to supply the lost stream (Section 3.3). In the Internet experiments, we will show that the overall quality is resilient to a view change and stays stable over the time. Recall that because we don't apply preemption in MC-n schemes, the number of victims will always be zero.

4.2 Internet Evaluation

We have integrated a prototype of the ViewCast model in our TEEVE testbed. Our goal is to observe the implementation feasibility of ViewCast, the performance of response time, and the impact of view change and network dynamics under the scale of Internet.

4.2.1 Network Characteristics

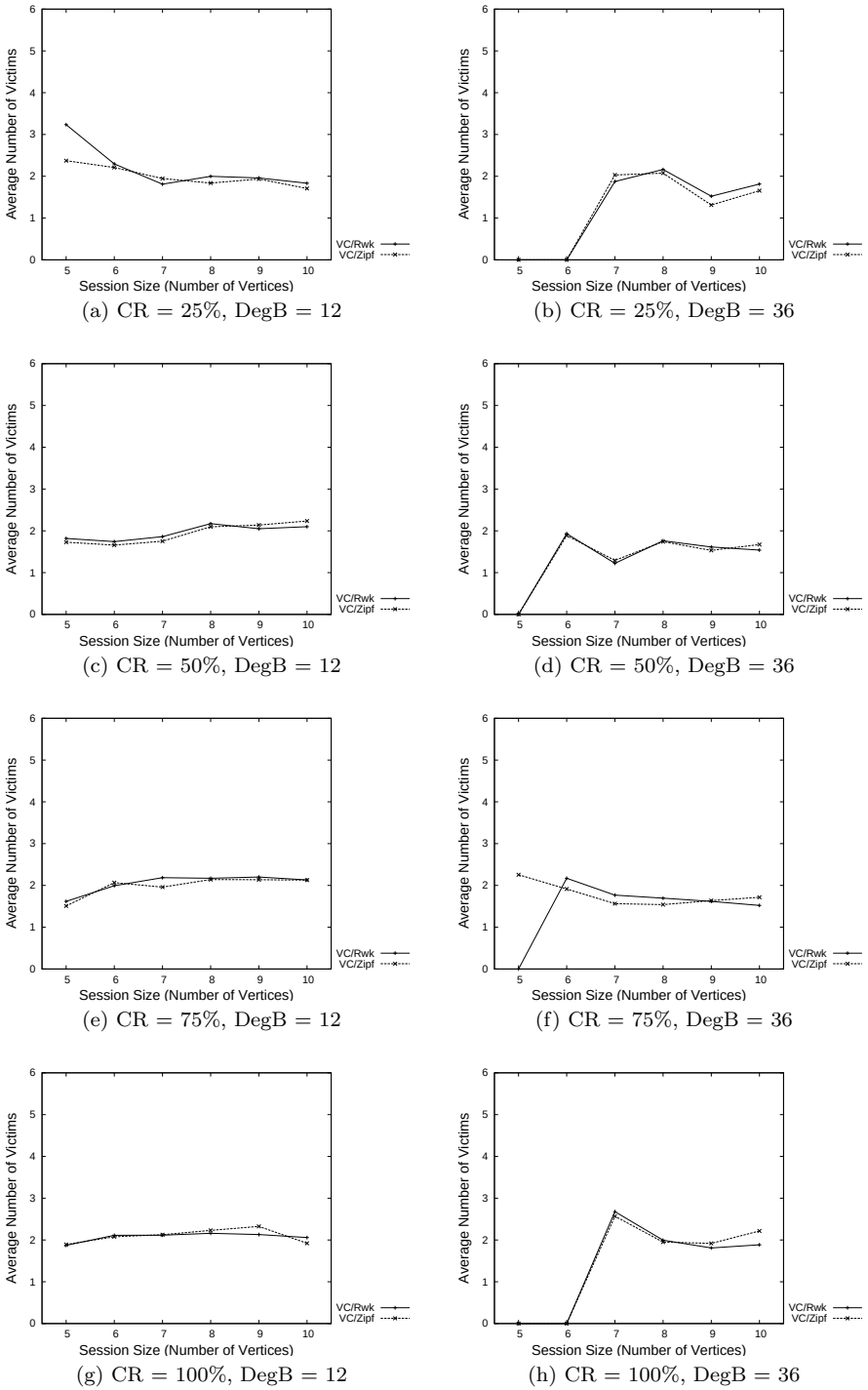
We currently have performed experiment using four 3DTI testing sites (Figure 15) including University of California at Berkeley (UCB), University of Illinois at Urbana-Champaign (UIUC), University of Central Florida (UCF), and National Center for Supercomputing Applications (NCSA). We construct 3DTI environments in UCB, UIUC and NCSA while including UCF only for testing purposes. We use 3D video streams recorded from remote Tai-Chi lectures and deploy them in those testing sites.

We set up one service gateway at each location. The session controller is located at the site of UIUC[5]. The connections among UCB, UCF, and UIUC run across the Internet2, while UIUC and NCSA are connected by the campus network. The network parameters are profiled and presented in Table III and Table IV.

We cannot extend the experiment to a larger scale because so far the four testing sites are

[4] As MC-n does not perform any preemption, there will be no victims of view change.

[5] The placement of the session controller has an impact on the system, for example, the response time of view requests. To ease the administration, we choose to place the session controller at UIUC.



(a) CR = 25%, DegB = 12
(b) CR = 25%, DegB = 36
(c) CR = 50%, DegB = 12
(d) CR = 50%, DegB = 36
(e) CR = 75%, DegB = 12
(f) CR = 75%, DegB = 36
(g) CR = 100%, DegB = 12
(h) CR = 100%, DegB = 36

Fig. 14. Average Number of Victims Per `fix_victim()`

all we have access to. We are unfortunately not able to use PlanetLab for experiments because our data rates would exceed the permitted quota. However, we argue that the scale of the 3DTI

system is not in the number of sites, but in the *streaming density* among multiple sites. It is also worth noting that the number of 3DTI sites is intrinsically limited by the visual complexity a user can accept.

Fig. 15. Distributed Multi-party 3DTI Testing Sites

4.2.2 Experiment Setup

The content of the 3D video streams has a large impact on the performance of the system, as the more complex 3D data incur more overhead. Therefore, we use four sets of "benchmark" pre-recorded 3D video streams, one at each site, to emulate the cameras for all experiments (except the one evaluating the 3D reconstruction performance). Each set consists of twelve 3D video streams capturing a person playing Tai-Chi in full-body movement.

In preparation, we first launch the service gateway at each site and start the session using pre-recorded 3D streams. Then users are invited to watch the Tai-Chi lecture. During the session, the user can rotate his/her view. The view change trace of each user is recorded so that we can perform more experiments with an automated program that follows the actual user view trace at each site simultaneously. The average view change degree is 33.1 ° with an average interval of 24.7 seconds.

Regarding to the set of priorities P (i.e. $\{p_1, p_2, ..., p_k\}$, refer to Section 3.2.1), we set $k = 6$. That is, for each site we sort all 12 streams in the order of the values coming from the differentiation function, select and assign priority to the top six streams.

4.2.3 Content and Computational Characteristics

The major computational overhead includes the time cost of 3D reconstruction, compression and rendering. All programs are written in C++. The 3D reconstruction is based on the trinocular stereo algorithm [Mulligan et al. 2001] and built with camera drivers provided by the Point Grey Research company for Windows systems. The compression is built using the cross-platform library of motion *JPEG* and *zlib*. The color and depth data are treated differently. The human visual system is relatively insensitive to variations in color. Compression algorithms such as MPEG and JPEG take advantage of this insensitivity by using lossy compression methods. We choose Motion JPEG to compress each frame separately, with no inter-frame motion estimation as in MPEG. This scheme sacrifices compression efficiency, but it eliminates the difficult problem of error propagation that may occur when frame packets are dropped. On the other hand, for the depth data any loss of depth information may distort the rendered volumetric image. Therefore, we decided to use the lossless algorithm of zlib to compress the depth information [Yang et al. 2006]. Finally, 3D rendering is implemented using OpenGL for both Linux and Windows systems. We use Dell 670 as the computing platform (two Xeon processors/dual-core/hyper-threading with 2GB memory).

Table V. Overall Data Rates of 3D Streams

Site	Rate
UCB	74.5 Mbps
UIUC	48.9 Mbps
NCSA	74.6 Mbps
UCF	60.9 Mbps

The profile of 3D reconstruction time is shown in Figure 16 based on 270 frames. The average time of 3D reconstruction is 106.9 ms. Throughout one 3DTI session, the whole system provides sustainable streaming for the collaborative virtual space with a rate of 7-10 macro-frames per second. It is worth noting that we are bounded by the camera hardware limit of capturing rate which is 15 frames per second. The 3D rendering time is below 100 ms per macro-frame on average [Yang et al. 2006] and the 3D compression/decompression cost is below 15 ms per 3D frame [Yang et al. 2006]. During the test based on pre-recorded 3D streams, we fix the frame rate at 10 frames per second. Using this replaying rate, Table V shows the aggregated data rate for twelve streams at each site, respectively.

4.2.4 Response Time

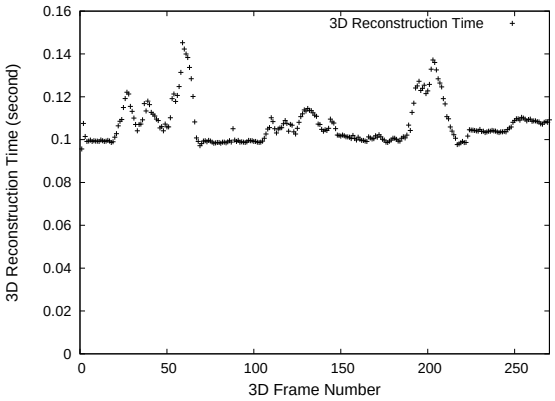


Fig. 16. 3D Reconstruction Time

We measure the response time of the ViewCast request. The response time is defined as $T_2 - T_1$, where T_1 is the time when the renderer triggers a view request to the local gateway, and T_2 is the time when the renderer receives all the streams from the other sites. We find that among the four sites, UC Berkeley has the maximal latency which is about 600 ms as shown in Figure 17. The average response time taken over the four sites for all requests is about 250 ms. The users report that they do not experience noticeable delay when changing the view. This reflects the advantage of the 3D data representation - the streams are independently retrieved and rendered, hence the visual quality is gradually improved as the contributing streams arrive.

4.2.5 Impact of View Change

Most importantly, we are concerned about the quality degradation incurred by view changes. The quality is measured using the *peak signal-to-noise ratio* (PSNR) metric, which is calculated

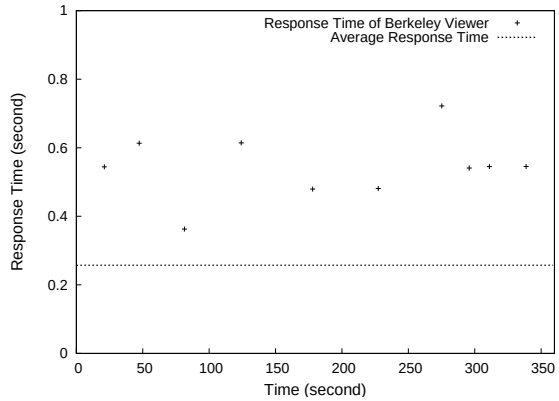


Fig. 17. Average and Maximal Response Time for ViewCast Requests

by comparing with the base case rendering of full content streaming (Equation 3),

$$PSNR = 20 \times \log_{10}\left(\sqrt{\frac{\sum_{i=1}^{height}\sum_{j=1}^{width}[I_{wa}(i,j) - I_{wo}(i,j)]^2}{height \times width}}\right) \quad (3)$$

where I_{wa} and I_{wo} denote the rendered 2D image using ViewCast and full content streaming respectively. The *width* and *height* denote the dimension of the rendered image.

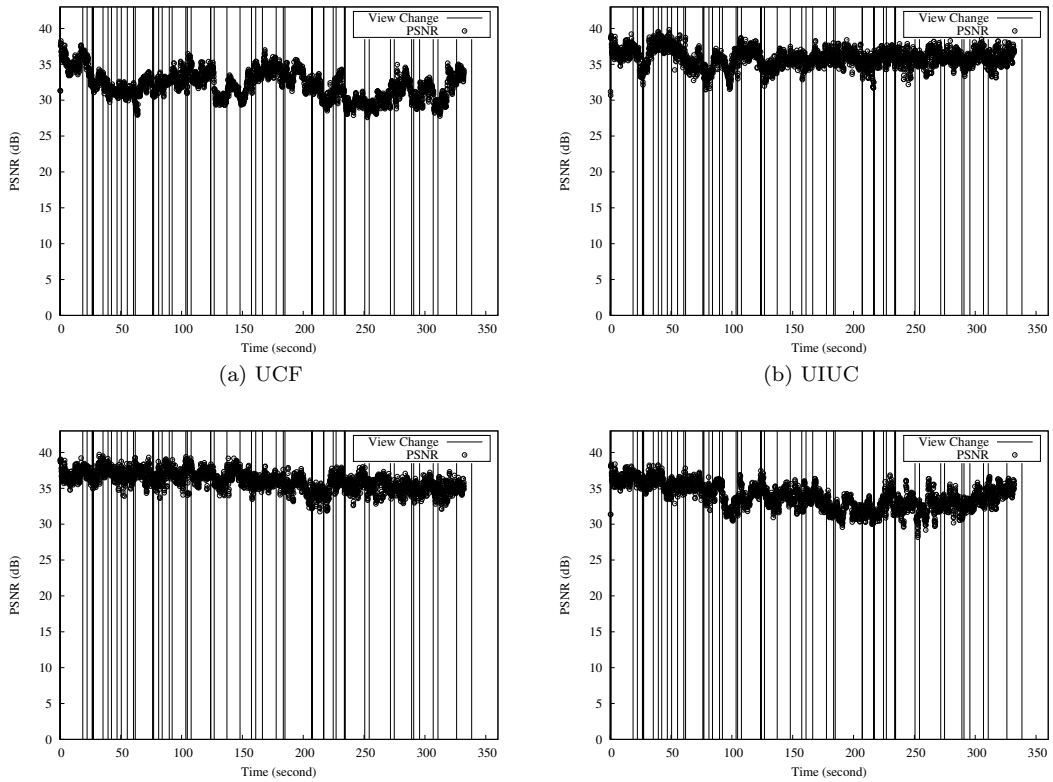
We aggregate all the view changes made by the users at four sites for the Tai-Chi video, and measure how the visual quality varies with the changes. Figure 18 shows the PSNR of individual frames from four rendering sites, respectively along with the view change instants of all users (marked by vertical lines). We note that the visual quality remains relatively stable between 30 and 40 dB for all sites. It is also interesting to note that when multiple users change views at the same time (where the view change lines become thick in Figure 18), the visual quality would typically have a dip. However, the variance is usually within the bound of 5 dB.

We also measure the total number of victims across four sites incurred by view changes. Figure 19 plots the results. It turns out that 88% of the view changes do not affect other vertices, i.e., the number of victims is 0. We observe that the view change made by users is often within small ranges. In the simulation, we artificially stress the system by allowing a large change of 360° range, where the number of victims is on average 2. With the real users, the majority of view changes yields no victims. There are sometimes 1-4 victims (about 12% in total), and our results for visual quality (Figure 18) demonstrates that the victims are quickly fixed and hence the overall quality does not vary largely.

4.2.6 Impact of Network Fluctuation

To evaluate the performance of ViewCast under the impact of the network fluctuation, we choose UCB as the target site and inject bi-directional traffic between UCB and other three sites. Figure 20 shows the effectiveness of the background traffic injection, which reduces the outbound throughput of ViewCast by around 30%. The inbound throughput is effected in similar way.

As we judge Figure 20, there are two interesting regions where the traffic injection occur: one from 52 seconds to 106 seconds where the average PSNR is high and the other one from 180 seconds to 236 seconds where the average PSNR is low. We want to examine how the network fluctuation would affect these two cases. The results are plotted in Figure 21 with traffic injection instants marked by vertical lines. To ease the comparison, we overlap the corresponding portion of Figure 18(d). As shown, the overall impact on actual quality is small (less than 5 dB). We also



(a) UCF (b) UIUC (c) NCSA (d) UCB

Fig. 18. Visual Quality

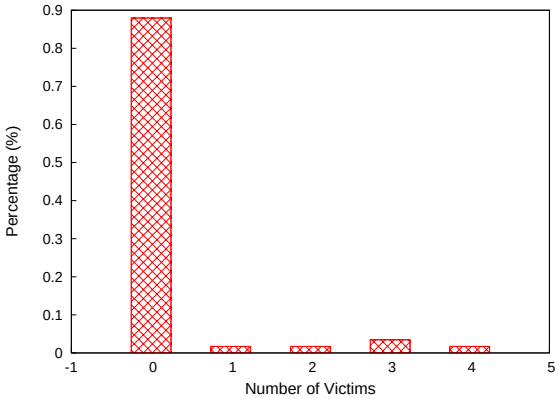


Fig. 19. The Distribution of View Changes (in percentage) with respect to the Number of Victims Generated

notice that the quality degradation turns out to be larger when the original PSNR was relatively large (about 36 dB in the first region) than when it was small (about 30 dB in the second region).

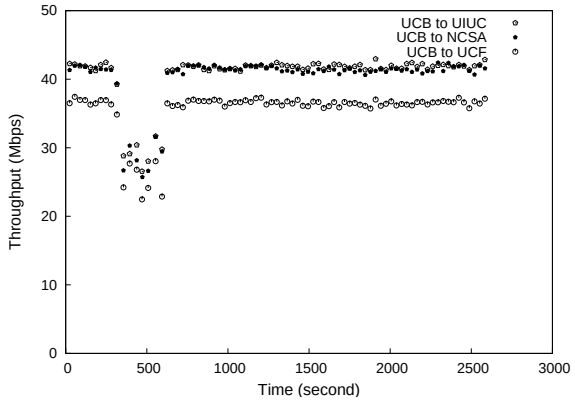


Fig. 20. Effectiveness of Background Traffic Injection

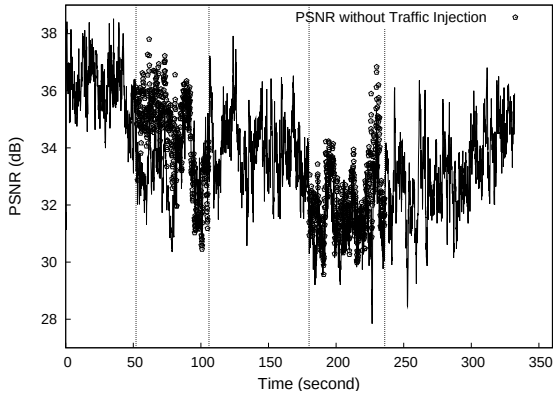


Fig. 21. Impact of Network Fluctuation on Rendering Quality

5. RELATED WORK

The most important application of the ViewCast concept is in the multi-party/multi-stream environment, which distinguishes it from the available protocols and techniques in several aspects.

Multicast protocols including application level multicast [Jannotti et al. 2000; Banerjee et al. 2002; Chu et al. 2000] are mostly concerned with the efficient transmission of particular stream(s) for a group of receivers. In contrast, ViewCast is a higher-level concept which focuses on the coordination of multi-streaming among multiple groups.

A coordination protocol for multiple streams was proposed to support tele-immersive systems [Ott and Mayer-Patel 2004]. However, the protocol only dealt with a pair of nodes with its main function of aggregating the information of each flow. The questions of how to apply it in 3DTI environments and how to interconnect multiple nodes were not addressed.

The 3D adaptation has been used in several tele-immersion systems and most of them take the view-based techniques. For example, Würmlin et al. implement a 3D video pipeline [Mürmlin et al. 2003] for the blue-c telepresence project [Gross et al. 2003]. The video system installs 16 CCD cameras covering 360° spatial range. During the runtime, 3 cameras are selected for the texture and 5 cameras for reconstruction based on the user view. The concern of adaptation is more focused on the 3D video processing and encoding part to make it affordable within resource limitations. However, the issue of QoS adaptation according to the user requirement and available

bandwidth, and the related spatial and temporal quality loss have not been addressed.

The related awareness driven model has been applied in collaborative virtual environments [Greenhalgh and Benford 1995; Reynard et al. 1998; Hosseini and Georganas 2003] for the Quality of Service (QoS) management. Given the awareness information of the user, the model dynamically selects the set of sources and the quality. Usually, each source represents one audio/video stream with multiple levels. However, the limitation of the model is its incapability of handling multiple correlated streams at each source and among sources as required in multi-party/multi-stream systems. For example, in [Hosseini and Georganas 2003] a multi-sender 3D videoconferencing application is implemented where a certain 3D effect is created by placing the 2D image of each participant in the virtual space. So each user is represented by one 2D stream. In their work, stream selection is used to reduce the downlink traffic of each user based on the orientation of the view and the *visibility*. Streams that are not considered as visible will not be sent to a particular user. The application of the stream selection is much simpler than the case of 3DTI environments where each user is represented by multiple streams.

6. CONCLUSION

This paper addresses the problem of supporting multi-party 3DTI environments using the novel ViewCast model. The main idea of the model is to take the view specification as the main goal for managing multiple correlated streams. Unlike previous systems which apply the stream-level QoS management, we are making the breakthrough by the argument that the QoS management in the multi-stream scenario should focus on the user view. The view-oriented approach not only provides more flexibility and adaptability but also a much more clear guidance to reach high-quality QoS management. Traditional content dissemination structures are focused on delivering a set of streams. There is almost none or very little coordination in the dissemination. For the first time, we claim that it is to the common benefit of all content consumers that coordination should be performed. Using the *view* as the high-level quality criterion, we can prioritize the delivery schedule to make it more efficient. We illustrate various aspects of the model using a real example of its application in a 3DTI environment. As shown in real experiments across the Internet and simulations, when combined with the application level multicast, ViewCast achieves low cost, efficient resource usage and high rendering quality. The experimental results encourage the possible usage of the model in other similar systems.

ACKNOWLEDGMENTS

This research is supported by the National Science Foundation (NSF SCI 05-49242, NSF CNS 05-20182, and NSF IIS 07-03756). The presented views are those of authors and do not represent the position of NSF. We would like to thank Dr. Peter Bajscy for his kind cooperation. We also thank the anonymous reviewers for their very helpful comments.

REFERENCES

3DAV Report 2003. Report on 3dav exploration. *International Organisation for Standardisation, ISO/IEC JTC1/SC29/WG11 N5878*.

Baker, H. H., Bhatti, N., Tanguay, D., Sobel, I., Gelb, D., Goss, M. E., Culbertson, W. B., and Malzbender, T. 2005. Understanding performance in coliseum, an immersive videoconferencing system. *ACM Trans. Multimedia Comput. Commun. Appl. 1,* 2, 190–210.

Banerjee, S., Bhattacharjee, B., and Kommareddy, C. 2002. Scalable application layer multicast. *SIGCOMM Comput. Commun. Rev. 32,* 4, 205–217.

Bauer, F. and Varma, A. 1997. Aries: A rearrangeable inexpensive edge-based on-line steiner algorithm. *IEEE Journal on Selected Areas in Communications (JSAC)*, 382–397.

Castro, M., Druschel, P., Kermarrec, A.-M., Nandi, A., Rowstron, A., and Singh, A. 2003. Splitstream: high-bandwidth multicast in cooperative environments. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*. ACM, New York, NY, USA, 298–313.

Chen, M. 2001. Design of a virtual auditorium. In *MULTIMEDIA '01: Proceedings of the ninth ACM international conference on Multimedia*. ACM, New York, NY, USA, 19–28.

Chu, Y. H., Rao, S. G., and Zhang, H. 2000. A case for end system multicast. In *SIGMETRICS '00: Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*. ACM, New York, NY, USA, 1–12.

Cui, Y. and Nahrstedt, K. 2003. Layered peer-to-peer streaming. In *NOSSDAV '03: Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video*. ACM, New York, NY, USA, 162–171.

Daniilidis, K., Mulligan, J., McKendall, R., Schmid, D., Kamberova, G., and Bajcsy, R. 2000. Real-time 3d-teleimmersion. In *Confluence of computer vision and computer graphics*. Kluwer Academic Publishers, Norwell, MA, USA, 253–265.

Gharai, L., Perkins, C., Riley, C., and Mankin, A. 2002. Large scale video conferencing: A digital amphitheater. In *Proceedings of the 8th International Conference on Distributed Multimedia Systems*. San Francisco, CA, USA, 137–143.

Greenhalgh, C. and Benford, S. 1995. Massive: a collaborative virtual environment for teleconferencing. *ACM Transaction on Compututer-Human Interaction 2,* 3, 239–261.

Gross, M., Würmlin, S., Naef, M., Lamboray, E., Spagno, C., Kunz, A., Koller-Meier, E., Svoboda, T., Gool, L. V., Lang, S., Strehlke, K., Moere, A. V., and Staadt, O. 2003. blue-c: a spatially immersive display and 3d video portal for telepresence. *ACM Transaction on Graphics 22,* 3, 819–827.

Hosseini, M. and Georganas, N. D. 2003. Design of a multi-sender 3d videoconferencing application over an end system multicast protocol. In *MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia*. ACM Press, New York, NY, USA, 480–489.

Jannotti, J., Gifford, D. K., Johnson, K. L., Kaashoek, M. F., and James W. O'Toole, J. 2000. Overcast: reliable multicasting with on overlay network. In *OSDI'00: Proceedings of the 4th conference on Symposium on Operating System Design & Implementation*. USENIX Association, Berkeley, CA, USA, 197–212.

Kauff, P. and Schreer, O. 2002. An immersive 3d video-conferencing system using shared virtual team user environments. In *CVE '02: Proceedings of the 4th international conference on Collaborative virtual environments*. ACM, New York, NY, USA, 105–112.

Mulligan, J., Isler, V., and Daniilidis, K. 2001. Trinocular stereo: a real-time algorithm and its evaluation. In *Proceedings of IEEE Workshop on Stereo and Multi-Baseline Vision, 2001. (SMBV 2001)*. Kauai, HI, USA, 10–17.

Mürmlin, S., Lamboray, E., and Gross, M. 2003. 3d video fragmens: dynamic point samples for real-time free-viewpoint video. In *Technical Report No. 397*. Institute of Scientific Computing, ETH, Zurich.

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

Raskar, R., Welch, G., Cutts, M., Lake, A., Stesin, L., and Fuchs, H. 1998. The office of the future: a unified approach to image-based modeling and spatially immersive displays. In *SIGGRAPH '98: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*. ACM Press, New York, NY, USA, 179–188.

Reynard, G., Benford, S., Greenhalgh, C., and Heath, C. 1998. Awareness driven video quality of service in collaborative virtual environments. In *CHI '98: Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 464–471.

Schreer, O., Brandenburg, N., Askar, S., and Trucco, E. 2001. A virtual 3d video-conferencing system providing semi-immersive telepresence: A real-time solution in hardware and software. In *Proceedings of the International Conference on eWork and eBusiness*. Venice, Italy, 184–190.

Shi, S., Wang, L., Calvert, K. L., and Griffioen, J. N. 2004. A multi-path routing service for immersive environments. In *CCGRID '04: Proceedings of the 2004 IEEE International Symposium on Cluster Computing and the Grid*. IEEE Computer Society, Washington, DC, USA, 699–706.

Sripanidkulchai, K., Maggs, B., and Zhang, H. 2004. An analysis of live streaming workloads on the internet. In *IMC'04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*. ACM, New York, NY, USA, 41–54.

Sriram, R., Manimaran, G., and Murthy, C. 1999. A rearrangeable algorithm for the construction of delay-constrained dynamic multicast trees. In *INFOCOM'99: Proceedings of the IEEE Conference on Computer Communications*. New York, NY, USA, 1073–1080.

TEEVE 2006. Teeve project, http://cairo.cs.uiuc.edu/teleimmersion, http://tele-immersion.citris-uc.org.

Yang, Z., Cui, Y., Anwar, Z., Bocchino, R., Kiyanclar, N., Nahrstedt, K., Campbell, R. H., and Yurcik, W. 2006. Real-time 3d video compression for tele-immersive environments. In *MMCN '06: SPIE Multimedia Computing and Networking*. San Jose, CA, USA, 12–23.

Yang, Z., Nahrstedt, K., Cui, Y., Yu, B., Liang, J., hack Jung, S., and Bajscy, R. 2005. Teeve: the next generation architecture for tele-immersive environments. In *ISM '05: Proceedings of the 7th IEEE International Symposium on Multimedia*. Irvine, CA, USA, 112–119.

Yang, Z., Wu, W., Nahrstedt, K., Kurillo, G., and Bajcsy, R. 2007. Viewcast: view dissemination and management for multi-party 3d tele-immersive environments. In *MULTIMEDIA '07: Proceedings of the 15th international conference on Multimedia*. ACM, New York, NY, USA, 882–891.

Yang, Z., Yu, B., Nahrstedt, K., and Bajscy, R. 2006. A multi-stream adaptation framework for bandwidth management in 3d tele-immersion. In *NOSSDAV '06: Proceedings of the 2006 international workshop on Network and operating systems support for digital audio and video*. ACM, New York, NY, USA, 1–6.

Received xxx 20xx; xxx 20xx; accepted xxx 20xx