# ViewCast: View Dissemination and Management for Multi-party 3D Tele-immersive Environments

Zhenyu Yang, Wanmin Wu, Klara Nahrstedt
University of Illinois at Urbana-Champaign
Department of Computer Science
201 N. Goodwin, Urbana, IL 61801, U.S.A.
{zyang2, wwu23, klara}@uiuc.edu

Gregorij Kurillo, Ruzena Bajcsy
University of California at Berkeley
Department of Electrical Engineering and Computer Sciences, 253 Cory Hall, Berkeley, CA 94720, U.S.A.
{gregorij, bajcsy}@eecs.berkeley.edu

## ABSTRACT
Real-time distributed multi-party/multi-stream systems are becoming more popular in many areas such as 3D tele-immersion, multi-camera conferencing and security surveillance. However, the construction of such systems in large scale is impeded by the huge demand of computing and networking resources and the lack of a simple yet powerful networking model to handle interconnection, scalability and quality of service (QoS) guarantees. We make two main contributions in the paper: (1) we propose a novel generalized ViewCast model for multi-party/multi-stream video-mediated systems that fills the gap between high-level user interest and low level per-stream management, and (2) we demonstrate the ViewCast model by applying it to the multi-party 3D Tele-Immersive (3DTI) collaboration among geographically dispersed users. More specifically, we show how the ViewCast model is used in supporting stream data dissemination, coordination and QoS management among multiple 3D tele-immersive environments. We present our experimental results in both real implementation and simulation to show that our ViewCast-based solution achieves high efficiency, scalability, and quality in supporting multi-party 3DTI collaboration.

## Categories and Subject Descriptors
C.2.1 [Network Architecture and Design]; C.2.2 [Computer Communication Networks]; C.2.3 [Network Operations]

## General Terms
Design, Algorithm, Performance, Experimentation

## Keywords
3D Tele-immersion, Network Protocols, Distributed Application, Coordination, Adaptation, Multicast

## 1. INTRODUCTION
With the continuing cost drop of digital cameras, real-time distributed multi-party/multi-stream systems are becoming more popular including 3D Tele-Immersive (3DTI) environments ([6][3][17]) and multi-camera surveillance systems ([1][7]). The application model of multi-party/multi-stream is shown in Figure 1, where each party is a site hosting multiple cameras that capture the local information represented by multiple video streams. Streams from different parties are then exchanged and aggregated through the network to provide a more comprehensive representation of a larger global environment as in the scenario of 3DTI environments, surveillance systems and video sensor network.

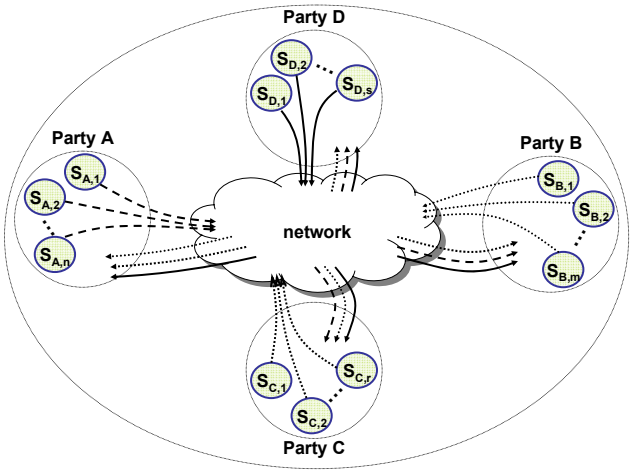


**Figure 1. Multi-party/multi-stream application model**

Our work is motivated by the goal of connecting multiple 3DTI environments to promote collaborative work among geographically distributed participants including education, entertainment, physical therapy, and artistic performance.

Most existing 3DTI systems only support two parties across the Internet. Supporting multi-party 3DTI collaboration is challenging due to the huge demand of networking and computing resources which strictly limits the system scalability. In 3DTI systems, each party represents one environment where an array of 3D cameras is installed from various angles to cover a wide field of view (>180°) of the local scene. When a person enters the environment, his/her complete 3D model is captured and represented by 3D video streams each generated from a 3D camera. With the help of a global coordinate system, the 3D representations of people from different remote places

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
*MM'07,* September 23–28, 2007, Augsburg, Bavaria, Germany.
Copyright 2007 ACM 978-1-59593-701-8/07/0009...$5.00.

can be merged and rendered together, creating a collaborative virtual space with a strong awareness of immersive experience for every participant (Figure 2).

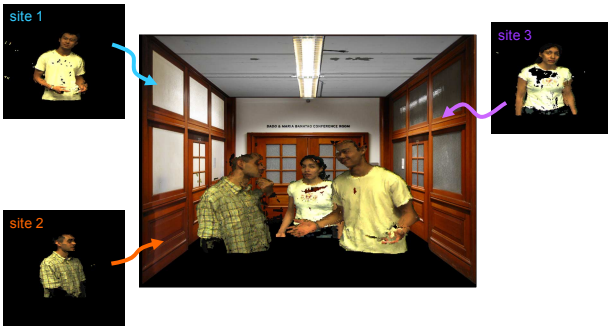


**Figure 2. Collaborative tele-immersive environments**

To achieve realistic 3D visual effect the rendering process usually requires multiple streams from each environment. In our experimental implementation (TEEVE, [17][20]), one 3D video stream has the basic rate over 30 Mbps (currently 320×240 frame resolution and 10 frames/second) to support interactive communication, and each environment produces up to 10 such streams. The image resolution is being upgraded to 640×480. If all streams are sent, the overall bandwidth from one environment will soon exceed Gbps level. The problem would become even more exacerbated if *N* such environments were connected. Meanwhile, the time cost of rendering one 3D image frame is about 13.7 ms on average. The rendering cost grows linearly with the number of streams. Although proposed 3D video compression techniques ([14][18]) have relieved the bandwidth and rendering requirement, advanced protocols of multi-stream coordination, QoS adaptation and network topology are needed to make more efficient use of computing and networking resources.

We take a novel approach by exploiting the user interest to reduce the resource consumption and provide quality guarantees. More specifically, we leverage an important and unique feature - *view* - in multi-party/multi-stream systems, which defines the field of interest or a viewing perspective that the users prefer to observe in the environment. For example, a view could be a viewpoint in 3DTI environment or a geographical point in surveillance system. To render a particular view, the system needs to select and transmit a subset of streams for compositing the view to the receiver.

We fill the gap between traditional networking models (e.g., multicast) formulated at the *stream level* and the support of view concept inherent in the *semantic level* by a novel generalized *ViewCast* model. The basic idea of ViewCast is that the user only specifies his/her view interest. The problems including how to map a view to specific streams and how to control multi-streaming are left to ViewCast. The insight is that with an ultimate goal of satisfying the rendering quality of particular view, the model ensures that the underlying transmission layer can have more flexibility in customizing streaming topology for improved multi-stream coordination and QoS adaptation.

We have applied the ViewCast model for joining multiple 3DTI environments in one collaborative session. Our solution to building multi-party/multi-stream system highlights the integration of ViewCast with an underlying application level multicast (e.g.,[10][15]) which is receiving increasing attention as an alternative to IP multicast. However, the process of constructing and maintaining ViewCast dissemination structure via multiple multicast trees becomes more complicated under our scenario. The key challenge is how to coordinate multiple multicast trees among streams and parties such that the resulting topology has the desired feature of QoS support, load balancing, high availability and resilience to view change.

We embed the ViewCast concept in the implementation of 3DTI infrastructure as a concrete example to illustrate the general ViewCast model and how it is used in building multi-party/multi-stream systems. Our collaborative virtual space currently has three major capturing/viewing 3DTI environments (with an average of 8 streams per site) and a few viewing sites connected across the Internet. The ViewCast-based distribution structure allows stable quality view rendering of the common virtual space. Meanwhile, extensive simulation tests indicate high efficiency, scalability and quality adaptation under larger scale of the 3DTI system.

In summary, our contribution is twofold: (1) we present for the first time a novel ViewCast model and (2) its application in enabling multi-party 3DTI collaboration. The remainder of the paper is organized as follows. In Section 2 we give an overview of 3DTI environments and present the ViewCast model. In Section 3 we then formalize the maximum quality and minimum quality problem in supporting multi-party 3DTI environments. Our solution is described in Section 4, and the evaluation results in Section 5. We finally discuss related work in Section 6, and conclude in Section 7.

## 2. OVERVIEW

We present an overview of the 3DTI environment (details in [17]) and the aspects related to the ViewCast model to facilitate further discussion.

### 2.1 3DTI Environment

There are three major tiers of 3DTI environment: 3D reconstruction, transmission and rendering. The 3D reconstruction tier is composed of multiple 3D cameras mounted at different angles in space. A 3D camera is a unit of four 2D digital cameras with one computer. The cameras are synchronized to take pictures at the same instant. The

pictures are then processed by the computer using trinocular stereo algorithm ([9]). The output 3D frame contains the color and coordinate information per pixel. Thus, each 3D camera produces one stream of frames corresponding to its angle. Since all cameras are capturing the same physical scene concurrently, the streams are semantically correlated. Their content constitutes a comprehensive representation of the scene. The transmission tier controls the streaming of 3D video across the network. The rendering tier takes the received streams from multiple sites and renders them in one common virtual space according to the current user view. Note that, since all cameras are calibrated with a global coordinate system, each of the streams can be independently rendered.

## 2.2 ViewCast Model

Generally, the *view* concept reflects the user interest at a higher-level, which distinguishes ViewCast from any networking model at lower stream-level. The ViewCast model basically specifies that when the user retrieves content from a multi-party/multi-stream system, only the user's view interest is required. The ViewCast model 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 environment. We envision the application of ViewCast in, for example, multi-camera conferencing, surveillance system, 3D TV, and video sensor networks. From the 3DTI prospective, 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.

- *Multi-party/multi-stream Environment*. The assumption for ViewCast is that each capturing source site supplies multiple correlated streams corresponding to a single view.

- *Semantic Stream Correlation.* The feature of semantic correlation is an important feature in multi-camera systems, where the concept of *view* has very intuitive definition.

- *Stream Differentiation.* Along with semantic stream correlation is the feature of stream differentiation. That is, a given view should favor some of the streams over others. In 3DTI environment, the rendering of 3D scene is view-dependent and the *contribution* of each stream to the rendering quality of the view could be different. As the angle of a 3D camera shifts away from the user view, its effective image resolution will decrease due to foreshortening and occlusion. As illustrated in Figure 3, given the user view (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 cameras are the least important.

- *Inter-stream Coding Independency.* We assume the coding/decoding independency among streams (i.e., we assume intra-stream coding as in [18]). 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 3, given the user view and the orientation of cameras ViewCast can select various subsets of streams (or cameras) such as {4}, {4,5}, {4,5,3} or {4,5,3,6}… depending on the quality and resource constraints. Inter-stream coding independency provides more flexibility in stream selection and QoS adaptation. Meanwhile, it also adds design challenge as there are more choices.

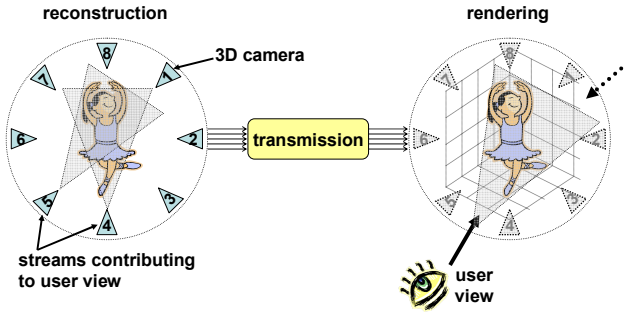


**Figure 3. Stream differentiation regarding to *view***

- *Open Model.* From an OSI layering point, the ViewCast model resides in the presentation layer. It maps between the semantics in the application layer and the stream manipulation in the session layer. More specifically, in 3DTI environments the ViewCast model only dictates desirable characters of the application (e.g., the definition of view and quality) and how these characters affect the multi-streaming. However, the design choice of higher and lower layers is open depending on specific requirements. We are the first to identify 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 view change. First, stream differentiation varies with view change. For example, in Figure 3 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 based on fixed stream differentiation such as layered coding and multi-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 is the discontinuity of rendering. If multicast protocol is used in the underlying layer, then stream switching at a parent node may impact the child nodes. The dynamics of view change presents a critical challenge for designing system based on ViewCast.

The key idea of ViewCast is that by leveraging the high-level view semantics the visual quality, which is closely related to view, can be guaranteed, while the low-level stream regulation layer can have larger flexibility for topology construction and QoS adaptation so that the resource constraints can be satisfied adaptively.

## 2.3 3DTI Session Architecture and Protocol

Figure 4 illustrates the 3DTI session architecture, which is managed at two layers. At the *local* layer, each 3DTI environment is managed by its service gateway (G), which consists of one or more computers. When a 3D camera (C) initiates, it registers with the service gateway to save the meta-data of its stream. Due to the runtime cost of 3D reconstruction, once a 3D frame is generated it is forwarded to the service gateway through high-speed LAN for further processing of data compression and streaming control. The gateway manages rendering as well and retrieves streams on behalf of its local renderers (R). Thus, service gateway is an application level data aggregating point at each 3DTI site. We implement ViewCast on top of the end system multicast ([15]) which is becoming an appealing alternative to IP multicast due to the advantage of flexibility and easy deployment. At the application level, service gateways collect multiple streams either from local sources (i.e., cameras) or from remote sources (i.e., peer service gateways). The collected streams are then multicasted according to the current status of the overlay network and the forwarding topology managed by the session controller (explained next). Therefore, the service gateway represents the node in the end system overlay network.

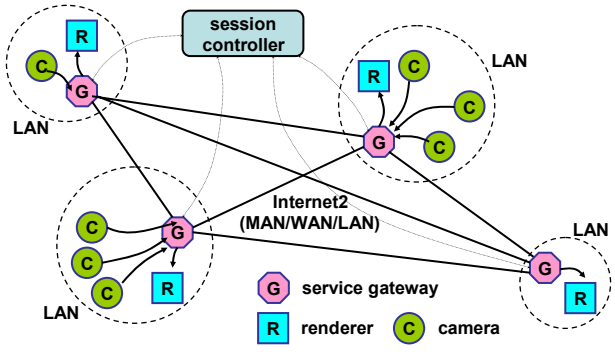


**Figure 4. An overlay of 3DTI session**

After the bootstrapping of local environment is completed, the service gateway registers with the central session controller at the *global* layer. The way session controller manages is similar to other proposed schemes (e.g., [11]). When a new service gateway joins the session, the session controller informs other service gateways to let them connect with each other and form the initial overlay graph, $G=\langle V, E\rangle$. Then the session controller starts to receive and serve view requests for each node and update the overlay structure accordingly (more details provided in Section 4).

For simplicity, it is assumed that all participating service gateways must register with the session controller before the live session can start.

During a live 3DTI session, the user can switch its viewing position of the virtual space via the input device (e.g., keyboard/mouse or head tracking) at the renderer. If the view change cannot be resolved, the renderer will forward it to the service gateway. The service gateway checks whether it has the streams available for accommodating the view change. Otherwise, it sends a view request to the session controller to compute a new multicast topology for coordinating the multi-streaming.

We take a centralized approach at the global layer because of its low messaging cost and responsiveness to the dynamics of 3DTI session. The approach is feasible in our situation where the number of service gateways is within a reasonable scale ($\leq 20$).

# 3. PROBLEM FORMULATION

We formalize the basic problems of ViewCast using 3DTI environment as an illustrative example. As shown in Figure 5, a node can request a view from a multi-stream source node. For example, node $v_2$ requests a view (denoted as $w_2$) from node $v_5$. Depending on the view and available resources, each requesting node may get different subset of streams. As long as the quality and resource constrains are satisfied, nodes which have available streams can serve other nodes. Furthermore, a node can retrieve streams from multiple nodes in parallel.

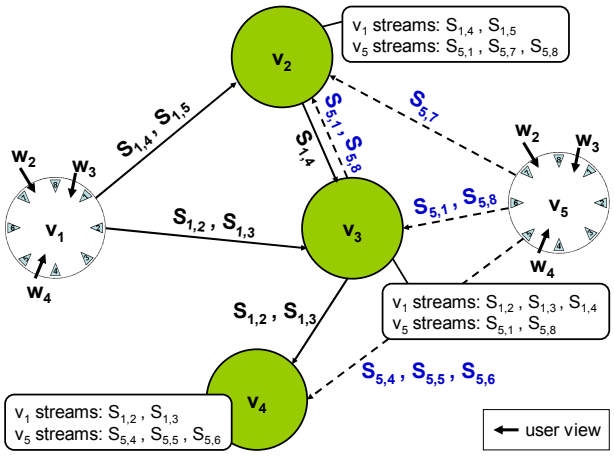


**Figure 5. ViewCast streaming**

We introduce the basic components of the ViewCast model based on the initial graph $G = \langle V, E\rangle$ as introduced earlier.

***Streams***: In the multi-party/multi-stream system, a node $v_i$ generates a set of streams. We denote $S_i$ as the set of streams originated from node $v_i$ (i.e., $S_i=\{s_{i,1}, s_{i,2},\ldots, s_{i,|Si|}\}$). Each stream $s_{i,j}$ ($1 \leq j \leq |S_i|$) has extra field-of-view (fov) information, denoted as $s_{i,j}.fov$ which represents the view

information of that stream. Note that, it is possible to have $S_i = \varnothing$ as the node may only serve rendering and viewing (i.e., no capturing cameras at the node $v_i$). The complete stream space is denoted as $S$ (i.e., $S = \bigcup_{\forall v_i \in V} S_i$).

***Nodes***: $V = \{v_1, v_2, \ldots, v_N\}$. There are $N$ nodes. In 3DTI environments, a node $v_i$ represents a service gateway which manages its local multiple streams. It can retrieve streams from other nodes as well by sending view request. Node $v_i$ is characterized by the following parameters.

- Inbound and outbound bandwidth constraints, denoted as $I_i$ and $O_i$ respectively. For simplicity, we assume all streams have the same data rate. Then $I_i$ and $O_i$ are degrees measured as the number of streams $v_i$ can receive and send. The inbound (and outbound) capacity is partitioned into 'reserved' pins, where each bin hosts one 3D video stream.

- Set of inbound streams $R_i$ where $R_i \subseteq S - S_i$. We denote $R_i(v_j)$ as the set of streams received which are originated from node $v_j$ (i.e., $R_i(v_j) \subseteq S_j$).
- Set of outbound streams $F_i$ where $F_i \subseteq R_i \cup S_i$. For stream $s \in R_i$, $s$ is called a *relay stream*. Otherwise, stream $s \in S_i$ is called *original stream*.

- Cost of stream. Consider a stream $s$ at node $v_i$ (i.e., $s \in R_i \cup S_i$). If $s \in R_i(v_j)$ and the related streaming path of $s$ from $v_j$ to $v_i$ is $p(v_j \xrightarrow{s} v_i)$, the cost of stream $C_i(s)$ is defined as,

$$C_i(s) = \sum_{e \in p(v_j \xrightarrow{s} v_i)} C(e) \qquad (1)$$

where $C$ is the cost function of edge (defined next). Otherwise, if $s \in S_i$ then $C_i(s) = 0$. The cost of stream reflects the delay from the source to the destination.

***Edges***: $E = \{\langle v_i, v_j \rangle \mid v_i \in V \wedge v_j \in V\}$. We define a cost function $C: E \rightarrow \Re$, which maps an edge to a real number. The cost function indicates the delay along the edge.

To summarize, for any node $v_i$ the following condition must always be satisfied.

$$R_i \subseteq S - S_i \wedge F_i \subseteq R_i \cup S_i \wedge |F_i| \leq O_i \wedge |R_i| \leq I_i \qquad (2)$$

For any node pair $(v_i, v_j)$ and stream $s$, if node $v_i$ forward stream $s$ to node $v_j$ directly through edge $\langle v_i, v_j \rangle$ then the following condition must be satisfied,

$$s \in F_i \wedge [C_i(s) + C(\langle v_i, v_j \rangle) \leq D] \qquad (3)$$

where $D$ is the delay bound for interactive communication. We name conditions (2) and (3) as the *system constraints*.

***View Model***: The abstract concept of *view* reflects the user interest in retrieving the content. We denote $w_{i,j}$ as the view request of node $v_i$ to $v_j$. In 3DTI environment, the view is represented by the direction and space of user-specified visible area. Objects inside the view are considered visible and rendered. The whole set of view requests is denoted as $W$ (i.e., $w_{i,j} \in W$).

***Differentiation Function***: The differentiation function is denoted as $D: S \times W \rightarrow \Re$, which gives the importance of a stream regarding to a given view and the fov of stream. The exact calculation of differentiation function can be found in [19]. If the environment is small enough, then the view can be simply represented as the direction vector in 3D space. Suppose the view request is denoted by the unit vector $\vec{w}$ and the direction of stream as the unit vector $s.\overrightarrow{fov}$. Then the differentiation function calculates the stream importance as in Equation (4).

$$D(s, \vec{w}) = s.\overrightarrow{fov} \cdot \vec{w} \qquad (4)$$

In above equation, the value of the dot product is $\cos\theta$ where $\theta$ is the angle between the vectors of $s.\overrightarrow{fov}$ and $\vec{w}$. When $s.\overrightarrow{fov}$ and $\vec{w}$ are close to each other, the dot product is close to 1, showing that stream $s$ is very important to the view. Otherwise, the value decreases to –1.

***Quality Function***. The quality function of rendering is denoted as $Q: 2^S \times W \rightarrow \Re$, which dictates that the rendering quality of view depends on the set of streams received. Sometimes it could be quite complicated to derive an exact form of quality function as in the case of 3DTI environment. A simple approach is taken as in Equation (5).

$$Q(S', \vec{w}) = \sum_{s \in S'} D(s, \vec{w}) \qquad (5)$$

Given above definitions, there is the *maximum quality problem* of ViewCast as formalized below.

**Maximum Quality Problem**

1. **to maximize** $\sum_{v_j \in V} Q(R_i, \vec{w}_{i,j})$
2. **subject to** system constraints (2) and (3)

The maximum quality problem is NP-complete, which can be proved based on the layered peer-to-peer streaming problem proposed in [16]. Another related problem of ViewCast is the *minimum quality problem* as given below.

**Minimum Quality Problem**

1. **to satisfy** $\forall v_i, v_j \in V, \ Q(R_i(v_j), \vec{w}_{i,j}) \geq \Delta$
2. **subject to** system constraints (2) and (3)

where $\Delta$ is a given lowest bound on acceptable quality. The minimum quality problem is also NP-complete as shown by studies on finding minimum-cost degree-constrained multicast trees ([4][12]).

In summary, we explain the basic idea of ViewCast under the 3DTI scenario. To simplify the explanation, certain

technical restraints are imposed on the model. However, those restraints are not inherent in the general ViewCast concept. For example, it is not required that all streams have similar data rate or quality. The essence of ViewCast includes the definition of view and whether it can be used to differentiate streams.

# 4. SOLUTION

There are two major goals for the ViewCast-based solution.

- *Minimum quality guarantee*: each node should receive a minimum set of streams to have some quality guarantee of every other node inside its view. For 3DTI environments, it implies the consistent presence of all participants in the virtual space, which is critical for collaborative work.
- *View change resilience*: when a node changes its view, the impact on other affected nodes should be minimized for the continuity of interaction.

## 4.1 Minimum Quality Guarantee

Because the minimum quality problem is hard, we propose heuristic using the priority-based approach ([19]) with the following steps.

**Step 1.** For the view request $\vec{w}$ and node $v_i$, the importance of streams is calculated using $D(s,\vec{w})$ for $s \in S_i$.

**Step 2.** The streams are selected against a threshold, for example, $D(s,\vec{w}) \geq 0$ (i.e., a 180° total view range).

**Step 3.** The selected streams are further differentiated into several priorities according to their importance. In 3DTI environments a node has around 8 streams. The stream selection in Step 2 produces a subset of 3 to 4 streams. We then define the set of priorities $P$ as $\{p_1, p_2, p_3, p_4\}$, where $p_4$ is the highest priority. Next, we assign priorities to selected streams according to $D(s,\vec{w})$. In our case, the stream having the largest value of $D(s,\vec{w})$ is assigned the priority $p_4$ and so forth.

**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 stream $s$ from node $v_i$ to node $v_j$. If both nodes have available bins, it is straightforward to establish the streaming. Otherwise, the bin of lower priority stream can be pre-empted. For example, if stream $s$ has $p_4$ priority based on the view, it can take the bin in either $v_i$ or $v_j$ occupied by streams of lower priority (i.e., $p_{1,2,3}$). When the pre-emption is needed, the bin of lowest priority stream will be taken first. The bin allocation of selected streams is performed in descending order of priority and terminated when the pre-emption is not possible.

Currently, we control the QoS of view rendering 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 rate of quality) in ViewCast.

## 4.2 View Change Resilience

Suppose node $v_i$ and $v_j$ have similar views and $v_j$ is streaming from $v_i$. When node $v_i$ changes its view, streams needed by $v_j$ may temporarily become unavailable. The impact will grow as the size of dependent nodes increases. As we notice, the view change operation is a frequent phenomenon in 3DTI environment and may cause large overhead if not treated properly.

Previous solutions rely on concepts of soft leave ([11]) and buffering ([16]). Soft leave requires the changing node to continuously serve old streams until affected nodes have found replacement. Although doable, under multi-stream scenario it would incur longer delay. Buffering let the intermediate nodes continue streaming from cache to absorb the propagation of quality degradation. However, it is not a feasible approach for live communication.

We apply two techniques to improve the view change resilience: *source diversification* and *stream bundling*. Source diversification attempts to diversify the supplying nodes to lower the dependency on each individual node. Meanwhile, stream bundling distributes streams of the same priority group among supplying nodes to even the importance of each node.

The basic idea is illustrated in Figure 6, where nodes $v_a$, $v_b$ have the same set of streams (i.e., $s_{1,2,3,4}$) that can be relayed to node $v_c$. The stream priority of $v_c$ as determined by its view (denoted as $w_c$) is $s_1$: $p_4$, $s_2$: $p_3$, $s_3$: $p_2$ and $s_4$: $p_1$. Under that, the streaming schedule on the right part is considered better since it provides a more evenly distribution of streaming quality among the sources. Note that, source diversification improves load balancing in a way similar to the SplitStream technique ([10]).

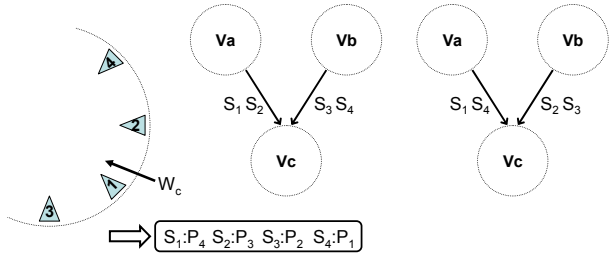


**Figure 6. Source Diversifying and Stream Bundling**

Finally, possible techniques at the application level can also limit the overhead of view change effectively. For example, in 3DTI environment most of the view change occurs with small degree which can be tolerated by human

perception. The rendering tier sends view request only when the view change is significant enough.

## 4.3 ViewCast Management

The main task of ViewCast management is to serve view request ($w_{i,j}$), which is performed by the session controller for each node ($v_i$). The main `serve_view` algorithm is sketched in Table 1 using notations introduced in Section 3.

**Table 1. View Management Algorithm**

```
serve_view(v_i, w_{i,j})
   S̄ ← get_streams(S, w_{i,j})
   for each s ∈ S̄ in descending order of priority
      v ← find_source (s, v_i)
      if (v == null)
         report rejection
      end
      out ← find_out_bin(s, v, v_i)
      in ← find_out_bin(s, v, v_i)
      if (out == null or in == null)
         report rejection
         return
      end
      serve_stream(s, v, out, v_i, in)
   end
   fix_victim()
end

find_source(s, dst)
   for each v ∈ V other than dst
      if (v can stream s to dst under system constraints)
         if (v has extra outbound bins)
            V1 ← V1 ∪ v
         else
            V2 ← V2 ∪ v
         end
      end
   end
   if (V1 ≠ ∅)
      return v∈V1 where v has the least forwarding to dst
   else
      return v∈V2 where v has pre-emptable bins
   end
end
```

The `get_streams` routine calculate the differentiation function $D(s,\vec{w})$ with the given view ($w_{i,j}$). The selected streams are assigned priority and saved in $\overline{S}$.

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

The `find_out_bin` and `find_in_bin` routines are pretty straightforward. They return either an unused bin or a bin used by lower priority stream for pre-emption. For selecting an outbound bin to be pre-empted, we prefer to choose the lowest priority stream. For selecting an inbound bin to be pre-empted, one important consideration is to select a stream that is least used in forwarding to reduce the pre-emption cost, because once an inbound stream is pre-empted all child nodes that rely on it will not be streaming from the parent node any more.

The `serve_stream` maintains the bookkeeping of inbound and outbound bins of source and destination nodes. When pre-emption is performed, the affected nodes are saved in the victim set.

The `fix_victim` routine tries to fix the broken link caused by pre-emption. To reduce the cost, this routine only fixes the broken link related to higher priority streams (e.g., $p_4$ and $p_3$). Otherwise, the affected node will simply ignore the lost stream and propagate the message to its child nodes. The propagation will terminate either when the pre-empted stream is not important for all child nodes or some child nodes have found new sources. The process is similar when the view change happens.

After the `serve_view` routine is completed, the session controller calculates the new topology and broadcasted to all nodes (i.e., service gateways).

# 5. EVALUATION[1]

We have integrated the ViewCast model in our 3DTI implementation which connects three major environments (UC Berkeley ↔ UIUC ↔ NCSA) and a few other viewing nodes across the Internet. Each major environment has up to 8 streams. Throughout one entire tele-immersive session, the whole system provides sustainable streaming of the common virtual space with an average frame rate of 10 frames per second (with the frame resolution of 320×240 pixels and the basic data rate of 30 Mbps). Figure

[1] We refrain from comparing our protocol with existing multi-cast protocol because the existing solutions are based on per-stream subscription, and ours solution regulates stream adaptation according to user request on view. It is thus very difficult to reach a fair comparison.

7 shows the variation in rendering quality measured in peak signal-to-noise ratio (PSNR) against the rendering with all streams following view changes. For smaller view changes, the most important stream usually resides in the inbound bins of the receiver. Therefore, the quality loss is relatively smaller. The reverse is true for larger view changes.

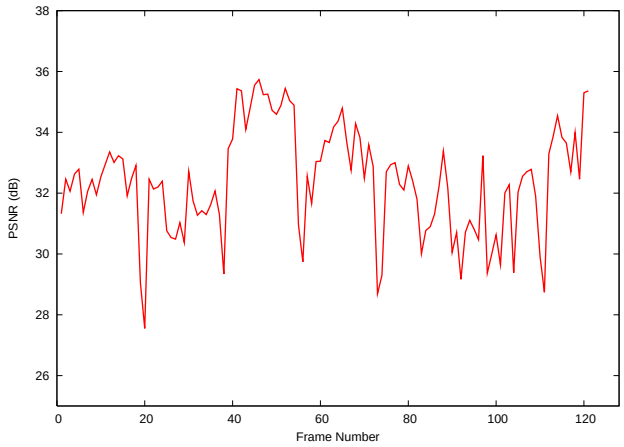


**Figure 7. Quality variation due to view changes**

For a more extensive evaluation, we simulate the ViewCast model in a 3DTI system of 4-20 nodes. For a fixed number of nodes, two hundred random configuration samples are generated to compute the average (and variance) of certain evaluation metrics. The configuration of each sample specifies, 1) the in-degree bound and out-degree bound for each participating node $v_i$, 2) the view $w_{i,j}$ of each node, and 3) the topology of underlying overlay network. More details are presented in the following subsections.

## 5.1 User Request Specification

We compare our solution with stream-based multicast protocols in terms of user request specification. In existing multicast protocols (e.g., [11]), users have to explicitly specify which streams they wish to subscribe from which sites. This per-stream request paradigm requires significant input from users which may be too tedious or error-prone. ViewCast, in contrast, allows users to specify subscription requests with only the view information per site. This high-level request paradigm frees users from specifying per-stream request one by one, and meanwhile achieving high degree of customization.

We use the total number of requests made at all sites as the metric to evaluate the ease of user request specification. In the simulation experiments, the session is heavily loaded, as each node always subscribes to all other sites in the system (i.e., the worst case). The number of requests users have to make in total if a stream-based multicast protocol were used is thus the total number of streams that have been acquired by the nodes in the system after running the ViewCast protocol. Figure 8 shows that even in the worst case, ViewCast requires much less user input than its conventional multicast counterpart.

## 5.2 System Quality

One important goal as described in Section 4 is to achieve minimum quality guarantee, that is, each request is satisfied by at least one important stream. We evaluate this minimum guarantee in terms of the rejection ratio defined as the ratio of the number of rejected requests and the total number of requests. A request is said to be rejected if and only if no single stream is provided to serve it. Again, the worst case (i.e., each site subscribes to all other sites) is evaluated. Figure 9 shows that near zero rejection ratio is achieved for all random samples.

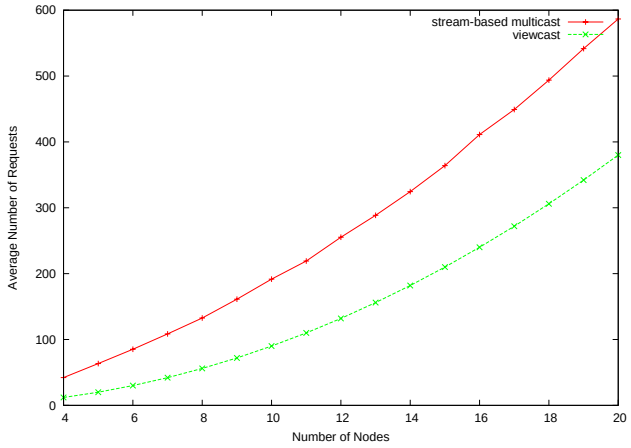


**Figure 8. User request specification**

Another important goal as described in Section 3 is to maximize the system quality subject to bandwidth constraints at each node. We define optimal quality as the quality without stream pre-emption. Figure 10 shows that our ViewCast protocol achieves near-optimal system-wide visual quality by the overlapping of the 'optimal quality' line and the 'achieved quality' line.

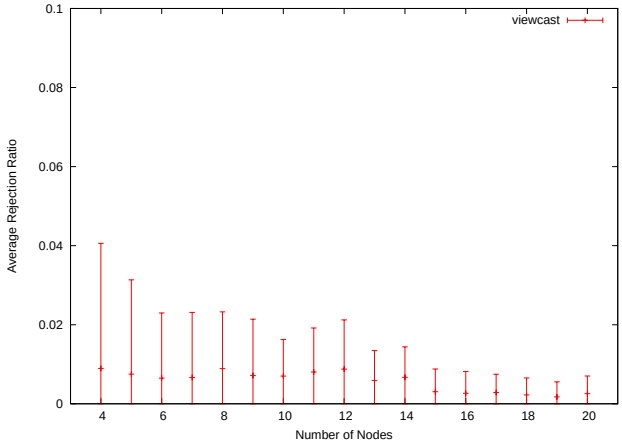


**Figure 9. Minimum quality guarantee**

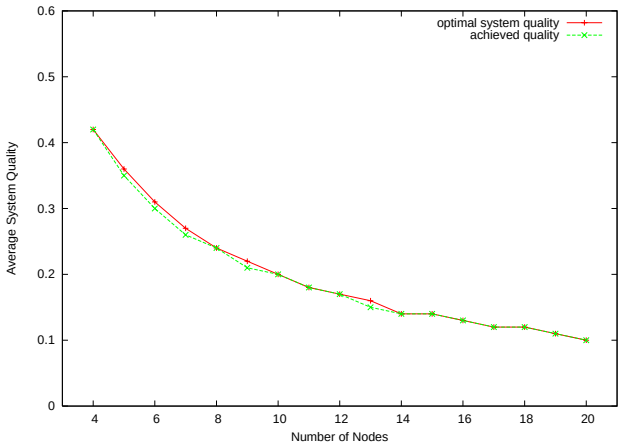


**Figure 10. Near-optimal and stable system quality**

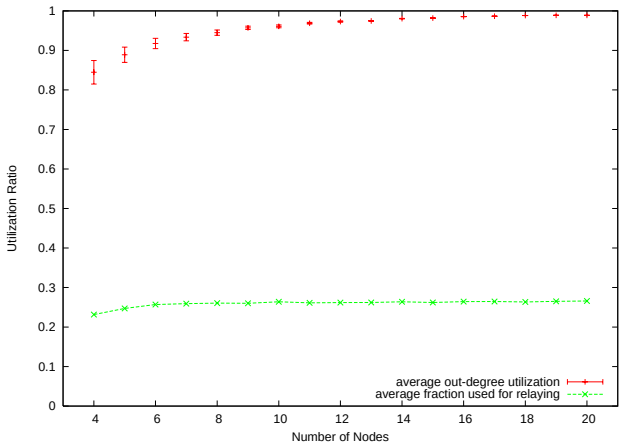


**Figure 11. Average out-degree utilization and Average fraction of out-degree for relaying**

## 5.3 Fairness

We ensure fairness among nodes in the system by load-balancing when selecting the forwarding node to serve an incoming view request. As in the previous experiments, the worst-case ViewCast session is used. We measure the average and variance of the out-degree utilization of nodes after the system overlay is constructed. This considers subscription requests at the initialization phase of a ViewCast session without taking the later view change into account. The out-degree utilization of a node is defined as the ratio of the used out-degree and the out-degree bound of the node. The average out-degree utilization as shown in Figure 11 is high and approaches 1.0 as the size of the system grows. Also note that it never exceeds 1.0 because we always satisfy bandwidth constraints at each node. The small variance of out-degree utilization as shown in Figure 11 demonstrates that the nodes in the system share roughly the same responsibilities of forwarding streams, and thus fairness is achieved.

We also evaluate how much of each node's out-degree is used to relay streams received from other nodes. Figure 11 further shows that consistently about a quarter of each node's out-degree is dedicated for forwarding streams. This indicates the strength of our multicast-based protocol that substantially reduces the burden of each source node compared to the conventional unicast solution.

## 5.4 View Change Cost

One way of observing the cost of view change is to exam the number of broken streams that need to be fixed. In the routine of `fix_victim` (Section 4), we keep track of the number of fixed streams and plot the data in Figure 12. The figure shows that the average number of fixed streams is quite small ($<2$) and gradually decreases when the number of nodes increases.

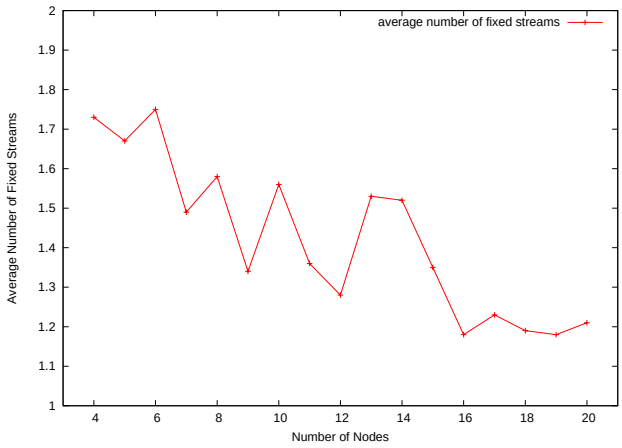


**Figure 12. Average number of fixed streams**

# 6. RELATED WORK

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

Multicast protocols including application level multicast (e.g., [8][11][13][15]) are mostly concerned with efficient transmission of particular stream for a group of receivers. In contrast, ViewCast is a higher-level concept which is focused on the coordination of multi-streaming among multiple groups.

Coordination protocol for multiple streams was proposed for supporting tele-immersive system ([3]). 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 awareness driven model has been applied in collaborative virtual environment ([2][5][11]) for quality of service 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.

## 7. CONCLUSION

We present the ViewCast model, a novel concept inside the presentation layer for constructing multi-party/multi-stream system in cooperative networking environment. The main idea of the model is to take the view specification as the main goal for managing multiple correlated streams. We illustrate various aspects of the model using a real example of its application in 3DTI environment. As shown in real experiments across the Internet and simulations, when combined with 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.

## 8. ACKNOWLEDGMENTS

We would like to acknowledge the support of this research by the National Science Foundation (NSF SCI 05-49242 and NSF CNS 05-20182). The presented views are those of authors and do not represent the position of NSF. We would also like to thank our colleagues in NCSA, Dr. Peter Bajscy and Miles Johnson, for their cooperation.

## 9. REFERENCES

[1] A. Girgensohn, F. Shipman, A. Dunnigan, T. Turner and L. Wilcox, Support for effective use of multiple video streams in security, In *Proceedings of ACM international workshop on Video surveillance and sensor networks (VSSN'06)*, 2006.

[2] C. Greenhalgh and S. Benford, Massive: a collaborative virtual environment for teleconferencing, In *ACM transactions on Computer Human Interactions*, 1995.

[3] D.E. Ott and K. Mayer-Patel, Coordinated multi-streaming for 3D tele-immersion, In *Proceedings of ACM international conference on Multimedia (MM'04)*, 2004.

[4] F. Bauer and A. Varma, Degree-constrained multicasting in point-to-point networks, In *Proceedings of IEEE conference on computer communications (INFOCOM'95)*, 1995.

[5] G. Reynard, S. Benford, C. Greenhalgh, and C. Heath, Awareness driven video quality of service in collaborative virtual environment, In *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI'98)*, 1998.

[6] H.H. Baker, N. Bhatti, D. Tanguay, I. Sobel, D. Gelb, M.E. Goss, J. MacCormick, K. Yuasa, W.B. Culbertson, and T. Malzbender, Computation and performance issues in Coliseum: an immersive videoconferencing system, In *Proceedings of ACM international conference on Multimedia (MM'03)*, 2003.

[7] J.-G. Lou, H. Cai and J. Li, A real-time interactive multi-view video system, In *Proceedings of ACM international conference on Multimedia(MM'05)*, 2005.

[8] J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek, and J. O'Toole, Overcast: Reliable multicasting with an overlay network, In *Proceedings of ACM symposium on Operating Systems Design and Implementation (OSDI'00)*, 2000.

[9] J. Mulligan and K. Daniilidis, Real-time trinocular stereo for tele-immersion, In *Proceedings of International conference on image processing*, 2001.

[10] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, SplitStream: high-bandwidth multicast in cooperative environments, In *Proceedings of ACM symposium on Operating systems principles (SOSP'03)*, 2003.

[11] M. Hosseini and N.D. Georganas, Design of a multi-sender 3D videoconferencing application over an end system multicast protocol, In *Proceedings of the eleventh ACM international conference on Multimedia (MM'03)*, 2003.

[12] R. Ravi, M.V. Marathe, S.S. Ravi, D.J. Rosenkrantz, and H.B. Hunt, Approximation algorithms for degree-constrained minimum-cost network-design problems, *Algorithmica*, 2001.

[13] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, Scalable application layer multicast, In *Proceedings of ACM Special Interest Group on Data Communication (SIGCOMM'02)*, 2002.

[14] S.-U. Kum, K. Mayer-Patel, and H. Fuchs, Real-time compression for dynamic 3D environment, In *Proceedings of ACM conference on Multimedia (MM'03)*, 2003.

[15] Y. Chu, S. Rao, and H. Zhang, A case for end system multicast, In *Proceedings of ACM Sigmetrics*, 2000.

[16] Y. Cui and K. Nahrstedt, Layered Peer-to-Peer Streaming, In *Proceedings of International workshop on network and operating systems support for digital audio an video (NOSSDAV'01)*, 2001.

[17] Z. Yang, Y. Cui, B. Yu, J. Liang, K. Nahrstedt, S.-H. Jung and R. Bajscy, TEEVE: the next generation architecture for tele-immersive environments, In *Proceedings of the 7th International Symposium on Multimedia (ISM'05)*, 2005.

[18] Z. Yang, Y. Cui, Z. Anwar, R. Bocchino, N. Kiyanclar, K. Nahrstedt, R.H. Campbell, and W. Yurick, Real-time 3D video compression for tel-immersive environments. In *Proceedings of SPIE multimedia computing and networking (MMCN'06)*, 2006.

[19] Z. Yang, B. Yu, K. Nahrstedt and R. Bajcsy, A multi-stream adaptation framework for bandwidth management in 3D tele-immersion, In *Proceedings of the 16th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'06)*, 2006.

[20] TEEVE project, http://cairo.cs.uiuc.edu/teleimmersion/, http://tele-immersion.citris-uc.org/