Dynamic Overlay Multicast in 3D Video Collaborative Systems

Wanmin Wu[†], Zhenyu Yang[‡]
[†]Department of Computer Science
University of Illinois at Urbana-Champaign
{wwu23, klara}@illinois.edu

Klara Nahrstedt[†]
[‡]School of Computing and Information Sciences
Florida International University
yangz@cs.fiu.edu

ABSTRACT

Multi-stream/multi-site 3D video collaborative systems are promising as they enable remote users to interact in a 3D virtual space with a sense of co-presence. However, the decentralized content dissemination remains a challenge. In this work, we explore approaches to construct adaptive overlay based on the users' visual interest in the collaborative space. Particularly, we consider the practical challenge that the user's interest might change dynamically. We propose, compare, and evaluate three algorithms to handle the view dynamics. With extensive experiments, we demonstrate that an algorithm that exploits view locality can achieve efficient bandwidth utilization, high topology stability, and great scalability.

Categories and Subject Descriptors

H.5.3 [**Group and Organization Interfaces**]: Computer-supported cooperative work; C.2.1 [**Network Architecture and Design**]: Network Topology

General Terms

Algorithms, Design, Performance, Experimentation

Keywords

overlay multicast, 3D video, collaborative environments

1. INTRODUCTION

The past decade has witnessed the emergence of 3D video collaborative (3DVC) systems [1, 6, 11]. By immersing users into a shared virtual space, these systems enable realistic human interaction among geographically distributed sites. Figure 1 depicts a three-site session where a doctor and two patients perform virtual physiotherapy. The technology finds other applications as well, such as gaming, remote training, consultation, and artistic performance.

Content dissemination for such systems is challenging. First, 3D video data consume much more bandwidth than their 2D counterparts; as the number of participating sites grows, smart data adaptation is a must. Second, 3DVC systems are usually *multi-stream/multi-site*, i.e., each participating site produces a number of data streams, so multiple overlays need to co-exist cooperatively among the participating nodes (sites) for full resource optimization. Third, smart data adaptation techniques are mostly dynamic; thus handling the dynamics well is crucial to the system stability.

Common to most 3DVC systems are the problems of (a) selecting data that are semantically important to transfer, and (b) selecting good overlay peers for data dissemination. The overall goal is to maximize networking resource utilization and stability while satisfying system constraints (e.g, bandwidth, latency). We tackle the problems in the context of 3D tele-immersive (3DTI) systems [11]. Specifically, we prioritize the 3D video streams based on user's visual Field of Interest (FoI) or view in the 3D collaborative space. The challenge that the user's view can be dynamically changing and thus affecting the overlay topology stability is the main focus of this work.

For dynamics handling, we explore new approaches to avoid stream disruption proactively, i.e., a peer that is less likely to lose the requested stream is considered better candidate as overlay neighbors. We propose, compare, and evaluate three algorithms: (a) *Random*, the simplest baseline algorithm that uses randomization for peer selection, (b) *Proximity First*, a locality-based algorithm that selects the peer that has the largest view proximity to the requesting node in the virtual space, and (c) *Priority First*, also a locality-based algorithm that selects the peer that has the highest priority for the requested stream. Our experiments show that Priority First performs better than the other two algorithms in terms of resource utilization, topology stability, and scalability.

Figure 1: Virtual physiotherapy.

NOSSDAV'09, June 3–5, 2009, Williamsburg, Virginia, USA.

2. RELATED WORK

We discuss the related research based on the characteriscs of 3DVC systems.

Multi-stream/multi-site streaming: The streaming of 3DVC systems often involves multiple sites, with each participating site being the source of multiple 3D data streams and also the receiver of many more streams from the other peers. Most existing peer-to-peer media streaming solutions focused on the topology construction for a single session/stream [2, 5]. The coordination among co-existent, competing streaming sessions was not considered. Ott *et al.* [7] studied the coordination of multi-stream delivery for tele-immersive systems. However, the protocol was designed for two-site collaboration. The interconnection and topology construction among multiple sites was not addressed.

Stream correlation: the streams produced from a tele-immersive site are semantically correlated. Hosseini *et al.* [6] studied the dynamic topology management in video-conferencing among peers, but they assumed the streams were independent and identical in terms of priority. We find that when the demand or stress for system resources is overly high, it is crucial to take a prioritized approach, where the limited resources are allocated first for the most important data.

Dynamics Handling: most existing work in overlay multicast targeted at maximizing resource usage (e.g., aggregate bandwidth) only [5, 6], whereas in P2P media streaming systems, there are works dealing with dynamics, but mainly for node joins/leaves (churn). In 3DVC systems, the peers availability is usually assumed to be stable, but the dynamics are caused by adaptation semantic changes. Moreover, the existing solutions to the dynamic problem (e.g., peer churn) are re-active, such as soft leave [6], buffering [4], and topology rearrangement [8]. We aim to minimize interference pro-actively. That is, we select a peer node that is less likely to lose the requested stream due to system dynamics. This approach is orthogonal to the existing re-active techniques, and can be combined with them to further reduce disruption.

3. SYSTEM MODEL

We introduce the system model and assumptions. Table 1 lists all the notations as a reference for the readers. We focus our discussion on the specific 3D tele-immersive applications, but the model and assumptions generally apply to other 3DVC systems as well.

Application Model. We consider the typical multi-stream/ multi-site applications in a 3DTI system, where a group of remote users collaborate in a 3D virtual space in real time. The collaboration requires everybody to see everybody else in the virtual world (Figure 1). We define the *cyberspace* as the 3D virtual space that contains all the *active* remote users in tele-immersive sites in the 3DTI system. There might be also viewing sites where *passive* remote users only join to watch the collaboration.

User Interest Model. In multimedia collaborative systems, there can be different modalities of user's interest or attention. For example, Yu explored the the audio FoI for distributed video lecturing systems [13]. In 3DVC systems, however, we consider the visual focus as the most important element to represent user interest. The majority of collaboration depends on the shared visual context among remote users [10]. In a 3D world, the vantage point of a user represents his/her FoI because it determines where the user's visual attention is directing. We denote v_i as a rendering viewpoint in the constructed cyberspace selected by the users at site i, which is mathematically a vector in the 3D coordinate world and can change dynamically. We assume the users at different sites can change views simultaneously. Further, we assume a view change made by the users can be continuous or discrete, but must be *finite*, that is, it must have a starting point and an ending point in a finite amount of time (e.g., Figure 2).

Overlay Network Model. $\{G_1, G_2, ..., G_N\}$ are the gateway nodes in all sites, where node G_i is in a site i. The gateways in tele-immersive sites are $\{G_1, G_2, ..., G_M\}$ $(M \leq N)$, and the gateways in viewing sites are $\{G_{M+1}, ..., G_N\}$ (if

Table 1: Notations and their definitions.

Notations	Definitions
G_i	Gateway node at site i, $1 \leq i \leq N$.
N	Number of participating sites.
M	Number of tele-immersive sites.
v_i	User view selected at site i.
S_i^L	Local streams produced at site i.
S	Set of all streams in the system
s_i^p or s	3D stream produced at site i with index p, $s_i^p \in S_i^L, 1 \leq p \leq \lvert S_i^L \rvert$.
$S_{v_i}^R$	Request stream set: the subset of streams G_i determines as the important streams to serve v_i. $\lvert S_{v_i}^R \rvert = N \times \kappa, S_{v_i}^R \subset S$.
$\vec{c_s}$	Camera view of stream s.
$f(v_i, s)$	$=\vec{v_i} \cdot \vec{c_s}$: contributing factor of stream s with respect to v_i.
κ	Number of streams selected from each site to serve a view.
p_j	Priority degree, $1 \leq j \leq \kappa$.
$\mathbf{P}$	$=< p_1, p_2, ..., p_\kappa >$: priority scale, the ordered set of priority degrees.
$\mathcal{P}(v_i, s)$	: $f(v_i, s) \mapsto p_j$ the mapping function from contributing factor to priority degree.
$C(v_i, s)$	The set of candidate nodes that has s to serve G_i for v_i, and also satisfies the latency constraint.
$C_1(v_i, s)$	The set of nodes that have available bandwidth to serve s to G_i, $C_1(v_i, s) \subseteq C(v_i, s)$.
$C_2(v_i, s)$	$= C(v_i, s) - C_1(v_i, s)$.
$I_j^{s'}$	Preemption impact at a candidate node for stream s'.

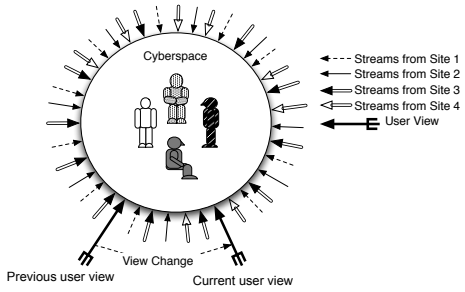


Figure 2: User view, view change, and camera views.

$M \neq N$). $G_i (1 \leq i \leq M)$ has a set of *local streams* produced by its local 3D cameras, denoted by S_i^L. Managed by a session controller for membership and topology information, the gateway nodes form an application-layer overlay for data dissemination. We envision a middleware functionality in the session controller to maintain an efficient overlay structure using one of the schemes described in Section 4[1].

Stream. Stream s_i^p is produced in a tele-immersive site i ($1 \leq i \leq M$), with p being the stream index within the site ($1 \leq p \leq |S_i^L|$). With the camera calibration parameters, we can map the capturing directions of the physical cameras from all sites into the 3D space. Each stream thus has a three-dimensional view vector, i.e., *camera view*, within this global space (Figure 2). The camera views are determined by the physical placement of the 3D cameras, and thus are static. The streams from one site are semantically correlated, because the cameras are shooting the same scene, only from different angles. Streams are differentiated by their camera views, $\vec{c_s}$ (Figure 2). We assume the streams are coded and transmitted independently.

3.1 Problem Description

In a nutshell, the general problem we are addressing is to construct an efficient topology for data dissemination among a network of N nodes, $\{G_1, G_2, ..., G_N\}$, in which M of them ($M \leq N$) are *active nodes* that actually produce data streams (S_i^L) continuously. The $M - N$ passive nodes do not produce data, but can be utilized to forward data, too.

Ideally, all the streams from each active node should be delivered to all other nodes so that a virtual space that contains all participants can be constructed. However, there are two constraints that need to be satisfied: the network bandwidth limits at each node and a soft real-time end-to-end delay bound between any pair of nodes. The goal is to maximize resource utilization while achieving topology stability subject to these constraints.

4. PROPOSED APPROACHES

We describe our approaches for the data dissemination problem in multi-stream/multi-site systems. In Section 4.1, we first present the protocols that are used to solve the two common problems in 3DVC data dissemination (Section 1), i.e., stream selection (based on semantic importance) and neighbor selection (to serve the selected streams). In Section 4.2 and 4.3, we present the specific algorithms to handle the dynamics of the system due to runtime adaptation.

4.1 Protocol Overview

Step 1. Stream Selection. Due to the stringent resource constraint, we prioritize the streams based on their *semantic importance*. In the conventional QoS techniques, the stream importance is static [4]. The priority of a stream in our approach accounts for the dynamics in the system, i.e., it varies with respect to (w.r.t.) the selected user view, and is thus more accurate. The stream importance is quantified by the metric *contributing factor*, defined as $f(v_i, s) = \vec{c_s} \cdot \vec{v_i}$ [12]. Finally, we assume each stream is coded and transmitted independently. Given a new view v_i selected by the user at site i, the gateway G_i examines all the streams in the system, $S = S_1^L \cup S_2^L \cup ... S_M^L$, w.r.t. this view, and selects *from each tele-immersive site* the top κ streams with the highest contributing factors w.r.t v_i. Large κ is good for achieving higher visual quality, but may increase the rejection ratio due to the limited bandwidth in the system. Smaller κ reduces rejection ratio, but may affect visual quality for the user. We find setting κ to $|S_j^L|/2$ (where $|S_j^L|$ is the total number of streams produced from G_j, $1 \leq j \leq M$) achieves fair balance between the two factors. The streams selected from each site are then sorted in the descending order of the contributing factors w.r.t. v_i, and the stream ranked j ($1 \leq j \leq \kappa$) is assigned the *priority degree* p_j of a priority scale. A priority scale $\mathbf{P} =< p_1, p_2, ..., p_\kappa >$ is an ordered set of p_i, where $p_j > p_{j+1}$ holds. We denote the priority mapping function as $\mathcal{P}_j(v_i, s) : f(v_i, s) \mapsto p_j$ where the contributing factor $f(v_i, s)$ is mapped to a priority degree. We denote the ordered set of streams as

$$S_{v_i}^R = \{\{s_1^1, s_1^2, ..., s_1^\kappa\}, \{s_2^1, s_2^2, ..., s_2^\kappa\}, ..., \{s_M^1, s_M^2, ..., s_M^\kappa\}\} \quad (1)$$

where streams $s_j^{\{1...\kappa\}}$ are those selected from site j ($1 \leq j \leq M$), as ordered in the descending order of priority. Thus, stream s_j^q has the priority degree p_q, i.e., $\mathcal{P}(v_i, s_j^q) > \mathcal{P}(v_i, s_j^{q+1})$ and $\mathcal{P}(v_i, s_j^q) = p_q$, where $1 \leq j \leq M$ and $1 \leq q < \kappa$. The insight of the stream selection process is that by differentiating the streams according to their semantic importance w.r.t. a view, we can efficiently utilize the limited resources by only delivering those semantically important streams.

Step 2. Parent Selection. For each selected stream, the gateway G_i checks whether it has the stream already. If so (e.g., those streams produced locally), the gateway delivers it to the renderer. Otherwise, a *subscription request* is generated in the form of $[view, streamID, priority]$ or $[v_i, s_j^q, p_q]$. A *priority queue* Q_q ($1 \leq q \leq \kappa$) is then constructed for each priority degree, and the set of requests are fed into the queue that has their priority degree in a random fashion (for fairness). For example, stream s_3^1 (stream # 1 from site 3) will be fed into queue Q_1, s_2^3 will be fed into Q_3, and so forth. Then the session controller finds the non-empty queue of the highest priority, removes a request $[v_i, s_j^q, p_q]$ from that queue, and looks for a "good" parent node to serve s_j^q using one of the approaches presented in Section 4.2. Note that we are using application-level overlay for the data dissemination, so any node (passive or active) that has (i.e., either produces or receives) s_j^q is a possible *candidate parent*. If such parent can be found, a reply is sent back to G_i, so that G_i can start connection with the parent node to receive s_j^q. Otherwise, rejection is reported.

4.2 Algorithms

We consider the following criteria when comparing the candidate nodes: (a) *bandwidth capacity*: the overlay link from the candidate node to the requesting node G_i should have sufficient bandwidth capacity to serve the stream; (b) *latency constraint*: if G_i were connected to the candidate node for receiving the stream, the end-to-end latency from the source of the stream to G_i should be smaller than a soft real-time bound; and (c) *chance of losing the stream*: we also consider the chance for the candidate node to lose the stream due to its own view change.

[1]We do not anticipate the scale of tele-immersive collaboration to grow beyond tens of nodes [9], so the simple and efficient centralized management approach is taken.

Next we discuss the details of several algorithms. We first introduce the following notations. (a) *Candidate Set*: $C(v_i, s)$, the set of nodes that has stream s to serve view v_i satisfying the latency constraint. (b) *Candidate Subset 1*: $C_1(v_i, s)$, the subset of $C(v_i, s)$ which have available bandwidth. $C_1(v_i, s) \subseteq C(v_i, s)$. (c) *Candidate Subset 2*: $C_2(v_i, s)$, the subset of $C(v_i, s)$ which do not have available bandwidth. $C_2(v_i, s) = C(v_i, s) - C_1(v_i, s)$. $C(v_i, s)$ addressed the latency constraint, and its subset $C_1(v_i, s)$ further resolves the bandwidth concern. We then propose three fundamental approaches, with further consideration of the reliability of a candidate node for serving the requested stream.

Random (Rand): Assuming no knowledge about the future and unpredictable user view change patterns, a natural and simple approach is to use a randomized algorithm. If there are candidates with available bandwidth (i.e., $C_1(v_i, s) \neq \emptyset$), randomly select one as the parent for G_i. If there is no such candidate but $C_2(v_i, s)$ is not empty, we select a candidate node G_c from $C_2(v_i, s)$ that has a lower-priority stream s' to preempt, that is, $\mathcal{P}(v_j, s') < \mathcal{P}(v_i, s)$. Further, among all such candidate nodes, we select one that has the minimum *preemption impact slot* (we defer the more detailed description of the preemption mechanism to Section 4.3). If no candidate is found, report rejection.

Proximity First (Pxf): This algorithm is based on the observation in our previous human experiments [10] that when users change view, they would usually change in small scale and thus stay in proximity relatively. We refer to this phenomena as *view locality*. Therefore, a candidate node G_c with a view v_c closest to v_i in the 3D space is likely to stay in the proximity (and thus have s) even when the user at site j changes her view. If $C_1(v_i, s)$ is not empty, for each node G_j in the set, compute its *view proximity* to v_i, as $\vec{v_j} \cdot \vec{v_i}$. Select the node with the highest view proximity to serve s. If there is no candidate with available bandwidth and $C_2(v_i, s)$ is not empty, use the same preemption mechanism as in Rand to preempt (Section 4.3). If both candidate sets are empty, report rejection.

Priority First (Prf): We further observe that a more reliable measure that captures the reliability of a candidate node for serving s is the priority degree of the stream to the node. That is, it is least likely for a node to lose a stream that is of the highest priority to it. If there are nodes with available bandwidth, sort the nodes of $C_1(v_i, s)$ in descending order of the priority of s w.r.t. its own view v_j. Select the node with the highest value of $\mathcal{P}(v_j, s)$ to serve G_i with s. If there is no candidate with available bandwidth but $C_2(v_i, s)$ is not empty, again use the preemption mechanism. If no candidate can be found, report rejection. Note that this algorithm is also based on view locality.

4.3 Preemption

As aforementioned, distributed 3DTI system has an overly high demand/stress for networking resources given the huge amount of data to deliver over COTS components. We believe it is important to take a prioritized approach in such context so that the limited resources are utilized efficiently. When no candidate has the spare bandwidth to serve s for G_i and s has a relatively high priority, we use the preemption mechanism to find a candidate node in $C_2(v_i, s)$ and preempt one of its lower-priority streams so that s can be delivered to G_i.

Unlike 2D video streams, 3D video streams are presented in an aggregated fashion on the renderers. Therefore, preempting some unimportant streams is much less noticeable to the users than in a 2D video-mediated system. For example, if the user is looking at a person's front view, losing some side stream is not as visually disrupting. The key is to identify the stream that is least important.

At preemption, we aim to minimize the impact to the downstream nodes in the overlay when performing preemption. So for each candidate node G_j in $C_2(v_i, s)$, we check if there is any stream s' the node is sending/forwarding that has a lower priority than the requested stream s, that is, $\mathcal{P}(v_j, s') < \mathcal{P}(v_i, s)$. If so, we compute the preemption impact for the slot/stream s', denoted by $I_j^{s'}$, as follows. Initially, $I_j^{s'}$ is set to 0. In the overlay tree of s', traverse the sub-tree rooted at G_j (excluding G_j). If the priority degree of s' to a downstream node G_h ($G_h \neq G_j$) is p_i, add $\kappa - i$ to $I_j^{s'}$. The final sum is the preemption impact for G_j on stream s'. Among the candidate nodes in $C_2(v_i, s)$ that has a lower-priority slot, select the node G_p with the minimum preemption impact out-slot, and use it to serve s to G_i instead.

If G_p is not a leaf node on the overlay tree for the victim stream s', we traverse the subtree of G_p to repair the disrupted nodes. Suppose s' has priority p_m to a downstream node G_d in the subtree rooted at G_p, we check whether $m \leq \kappa/2$, that is, if it is one of the more important streams. If so, a new request $[v_d, s', \mathcal{P}(v_d, s')]$ is generated and placed into the corresponding priority queue Q so the lost stream can be retrieved.

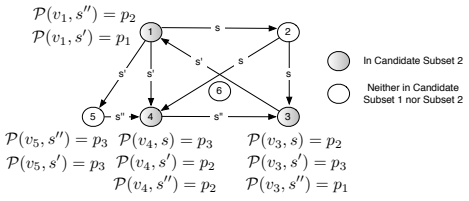


Figure 3: Example of preemption impact.

Figure 3 illustrates the preemption algorithm with an example. Assume s is the stream requested by G_6, and $\kappa = 4$. As shown in the figure, $G_1, ..., G_4$ are the nodes that have s. Suppose the latency requirement eliminates G_2 as a candidate. G_1, G_3, and G_4, however, do not have any spare bandwidth left to serve s, so they fall into $C_2(v_6, s)$. Assume $\mathcal{P}(v_6, s) = p_1$, meaning s is among the most important streams to G_6. Among the candidate nodes, G_3 has a lower-priority stream s', and G_4 has lower-priority streams s''. We can compute the preemption impact as $I_3^{s'} = 3(G_1) + 2(G_4) + 1(G_5) = 6$ and $I_4^{s''} = 3(G_3) = 3$. Hence, G_4 has the minimum preemption impact slot of s''. We thus preempt s'' and use this out-slot to serve s to G_6.

5. PERFORMANCE EVALUATION

5.1 Simulation Setup

We study the algorithms with extensive simulation. Below we describe the topology, node resources, and dynamic

Table 2: Number of Nodes Distribution

Total	Tele-immersive	Super Viewing	Normal Viewing
22	5	7	10
28	5	8	15
34	5	9	20
40	5	10	25
46	5	11	30

workload for the simulation setup. The simulator is written in C/C++.

Topology. We vary the total number of sites (or nodes) from 22 to 46, and differentiate three types of nodes in the topology: (1) *tele-immersive nodes*: which have original streams to send out, representing the gateways in 3D tele-immersion sites that have cameras installed, (2) *super viewing nodes*: which have direct connection (on the overlay level) to the tele-immersive nodes and available bandwidth to serve M views without any loss (i.e., $\kappa \times M$ streams), and (3) *normal viewing nodes*: which only have direct connection with the super viewing nodes. We evaluate five configurations (in terms of number of nodes) as shown in Table 5.1. The delay along each link is normal distribution of $N(50, 10)$, (i.e., average 50 ms). The maximum tolerable delay is 200 ms from the source to the destination.

Node Resources. Each tele-immersive node produces 8 streams, among which at most 4 streams should be selected to serve a view, i.e., $\kappa = 4$. Each stream has the data rate of 1 (unit). We evaluate two types of bandwidth distribution: (a) *uniform*: each link (among super and normal vertices) has the same bandwidth bound from 2 to 5 where the number means how many streams can be transmitted, and (b) heterogeneous: around 70% of the links have 3 or 4, and the rest have 2 or 5.

Dynamic Workload. We evaluate two types of view change workload: (a) *uniform* - which simulate the user data we observe [10]: the view change interval is a normal distribution of $N(60, 10)$, (i.e., average 60 seconds). The view change pattern is 95% of the time, a random walk with $N(35°, 5°)$, and 5% of the time, a more dramatic view change of 90° is applied. (b) *Zipf* - which is common pattern in data selection [3]: specifically ten view directions are pre-defined to uniformly divide the cyberspace with an alpha being 1.0, and the view change follows the distribution Zipf(10, 1.0) (i.e., the exponent is 1.0). Each simulation runs for 100 virtual minutes. Each run is executed 6 times to compute the confidence interval.

5.2 Rejection Ratio

Figure 4(a)(d)(g) show the overall rejection ratios for different algorithms when the bandwidth bound (BW) is 2, 5, and heterogeneous (denoted by HT), and when the workload follows normal distribution. The values are averaged across different runs. Due to the limited space, we do not show graphs for BW= 3, 4, which actually look quite similar to that of BW=5.

First, we observe that the locality-based approaches generally perform better than the randomized one. For example, when the total number of nodes is 46 and BW is 2, Prf and Pxf are roughly 33.3% better than Rand. With the same view change workload, utilizing view locality does significantly reduce future rejections. Second, the rejection ratio of Prf is 7%-10%, which is reasonable (confirming the choice of κ).

Since different streams have different semantic importance, we also evaluate the breakdown of the rejected requests. Figure 4(b)(e)(h) show the results. On average, the rejection ratio of the streams with p_1 is around 1%. Considering the overall rejection ratio is 8∼9% and there are four priority degrees in total, the margin of improvement is about 50%.

Due to the limited space, we do not show the graphs for the Zipf-distributed workload, which look quite similar to Figure 4(b)(e)(h). For example, when the total number of nodes is 46 and BW is 5, Prf and Pxf are roughly 35.3% better than Rand for Zipf-based workload. Prf and Pxf also perform similarly in terms of rejection ratio, but the overall rejection ratio of Prf is 5%-9%, which is a little lower than in the uniformly distributed data.

5.3 View Change Tolerance

We evaluate the system interference in terms of the number of victims incurred in each algorithm. The victim is defined as the victim stream that gets preempted for a high-priority stream. Figure 4(c)(f)(i) present the results.

First, although in terms of rejection ratio Pxf performs better than Rand, we see that Rand has lower number of victims than Pxf. For example, when the total number of nodes is 46 and BW is heterogeneous, Rand has about 16.7% less interference than Pxf.

Second, we observe that Prf performs much better than Rand and Pxf generally. When the size of the system is 46 and BW is 2, for example, Prf is about 46.7% better than Rand and 50% better than Pxf. And the trend is that as the number of vertices increases, the difference between the performances of Prf and Pxf or Rand increases.

For Zipf-based data, Prf and Pxf generally achieve lower numbers of victims than Rand. For example, when the total number of nodes is 46 and BW is 5, Rand has about 64% more victims than the two other algorithms. Prf is still generally better than Pxf, but the discrepancy (1%-10%) is smaller than that in the uniformly distributed workload.

6. CONCLUSION

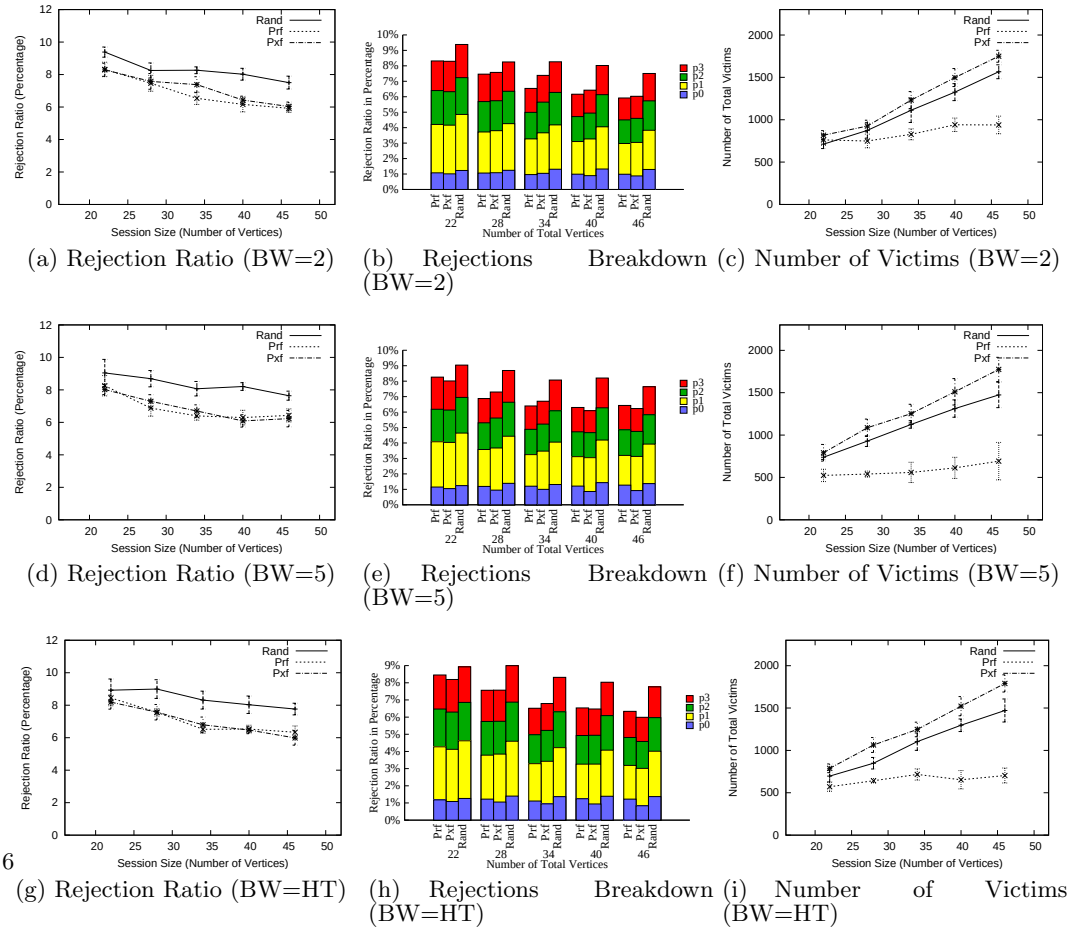
Our contribution in this paper is twofold: (1) we identify the challenges of dynamic topology maintenance in distributed 3DVC systems; (2) we compare three algorithms and demonstrate that Priority First achieves efficient resource utilization and high tolerance under different dynamics patterns by exploiting view locality in a fine granularity. Our future plans include collecting large amount of user traces and testing other 3DVC applications with wider scale of deployment on the Internet.

7. ACKNOWLEDGEMENT

This research is supported by National Science Foundation (NSF) under grants NSF IIS 08-40323, CNS 05-20182, CNS 07-20702, CNS 08-34480. The presented views are those of authors and do not represent the position of NSF.

8. REFERENCES

[1] H. H. Baker, N. Bhatti, D. Tanguay, I. Sobel, D. Gelb, M. E. Goss, W. B. Culbertson, and T. Malzbender. Understanding performance in coliseum, an immersive



(a) Rejection Ratio (BW=2) (b) Rejections Breakdown (BW=2) (c) Number of Victims (BW=2)

(d) Rejection Ratio (BW=5) (e) Rejections Breakdown (BW=5) (f) Number of Victims (BW=5)

(g) Rejection Ratio (BW=HT) (h) Rejections Breakdown (BW=HT) (i) Number of Victims (BW=HT)

Figure 4: Experimental Results

videoconferencing system. *ACM TOMCCAP*, 1(2):190–210, 2005.

[2] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. SplitStream: High-bandwidth multicast in cooperative environments. In *Proc. of ACM SOSP*, pages 298–313, 2003.

[3] M. Chesire, A. Wolman, G. M. Voelker, and H. M. Levy. Measurement and analysis of a streaming media workload. In *Proc. of USITS*, pages 1–12, 2001.

[4] Y. Cui and K. Nahrstedt. Layered peer-to-peer streaming. In *Proc. of ACM NOSSDAV*, pages 162–171, 2003.

[5] M. Hefeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava. Promise: peer-to-peer media streaming using collectcast. In *Proc. of ACM MM*, pages 45–54, 2003.

[6] M. Hosseini and N. D. Georganas. Design of a multi-sender 3D videoconferencing application over an end system multicast protocol. In *Proc. of ACM MM*, pages 480–489.

[7] D. E. Ott and K. Mayer-Patel. An open architecture for transport-level protocol coordination in distributed multimedia applications. *ACM TOMCCAP*, 3(3), 2007.

[8] S. Raghavan, G. Manimaran, C. Siva, and R. Murthy. A rearrangeable algorithm for the construction of delay-constrained dynamic multicast trees. *IEEE ToN*, 7:514–529, 1999.

[9] W. Wu, Z. Yang, I. Gupta, and K. Nahrstedt. Towards multi-site collaboration in 3d tele-immersive environments. In *Proc. of ICDCS'08*, pages 647–654, 2008.

[10] W. Wu, Z. Yang, and K. Nahrstedt. A study of visual context representation and control for remote sport learning tasks. In *Proc. of ED-MEDIA*, 2008.

[11] Z. Yang, K. Nahrstedt, Y. Cui, B. Yu, J. Liang, S. hack Jung, and R. Bajcsy. TEEVE: The next generation architecture for tele-immersive environments. In *Proc. of IEEE ISM*, pages 112–119, 2005.

[12] Z. Yang, B. Yu, K. Nahrstedt, and R. Bajcsy. A multi-stream adaptation framework for bandwidth management in 3d tele-immersion. In *Proc. of ACM NOSSDAV*, pages 1–6, 2006.

[13] B. Yu. *MyView: Customizable Automatic Visual Space Management for Multi-STream Environment*. PhD thesis, University of Illinois at Urbana-Champaign, 2006.