Towards Multi-Site Collaboration in 3D Tele-Immersive Environments

Wanmin Wu, Zhenyu Yang, Indranil Gupta, Klara Nahrstedt
Department of Computer Science
University of Illinois at Urbana-Champaign
{wwu23, zyang2, indy, klara}@uiuc.edu

Abstract

3D tele-immersion (3DTI) has recently emerged as a new way of video-mediated collaboration across the Internet. Unlike conventional 2D video-conferencing systems, it can immerse remote users into a shared 3D virtual space so that they can interact or collaborate "virtually". However, most existing 3DTI systems can support only two sites of collaboration, due to the huge demand of networking resources and the lack of a simple yet efficient data dissemination model. In this paper, we propose a general publish-subscribe model for multi-site 3DTI systems, which efficiently utilizes limited network resources by leveraging user interest. We focus on the overlay construction problem in the publish-subscribe model by exploring a spectrum of heuristic algorithms for data dissemination. With extensive simulation, we identify the advantages of a simple randomized algorithm. We propose optimization to further improve the randomized algorithm by exploiting semantic correlation. Experimental results demonstrate that we can achieve an improvement by a factor of five.

1 Introduction

3D tele-immersion is a new video medium that creates 3D photorealistic, immersive, and interactive collaboration among geographically dispersed users. Over the last few years, 3DTI has shown great potentials in a wide range of applications such as video-conferencing, distance learning of physical activities, collaborative art performance, firefighter training, and medical consultation [2, 12, 17, 24, 28].

As shown in Figure 1, each 3DTI site consists of an array of 3D cameras and an array of 3D displays. The 3D cameras are set up to capture the participant in the local scene from various angles, with each camera producing a continuous 3D video stream. The streams from all sites are exchanged through the Internet and aggregated at each 3D display in real time, such that an integrated 3D virtual space ("cyberspace") can be constructed that immerses the participants from all sites, as illustrated in Figure 2. The shared visual context enables remote users to interact or collaborate "virtually" in the cyber-space.

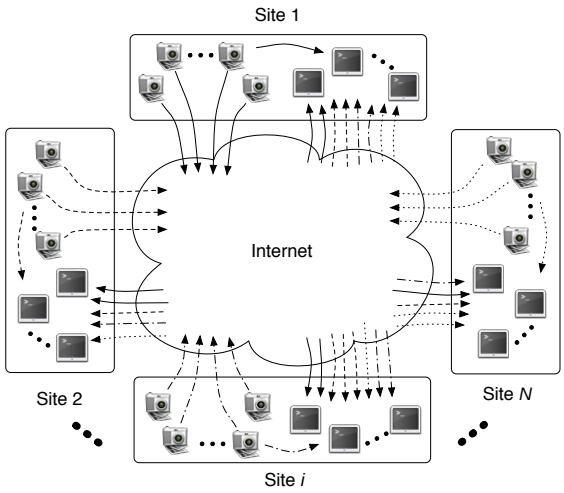


Figure 1. Multi-Site 3DTI Systems

However, the huge demand of networking and computing resources has restricted current 3DTI systems to work with only two sites. Each 3D video stream can consume a large amount of bandwidth (e.g., $640 \times 480 \times 15fps \times 5B/pixel \approx 180Mbps$), making even two sites of collaboration challenging enough. Moreover, the rendering time cost, which is about 10ms/stream by our measurement [28], grows linearly with the number of streams. The problem is exacerbated if multiple sites are connected together, with each site producing tens of such large streams, which can easily exceed the bandwidth and timing bounds[1].

Therefore, the "all-to-all" data distribution scheme, adopted by existing 2D video-conferencing and 3DTI systems ([2, 9]), has to be abandoned as the scale of the 3DTI system grows. As a concrete example, the 3DTI system described in [28] involved two sites thousands of miles apart,

[1] The timing bound is necessary to guarantee interactivity.

each sending about ten streams to the other. With the measured resource limits, even three-site collaboration was not possible if all streams from each site were sent to all other sites.

Some previous work has addressed the issues in a piecemeal manner: these include background subtraction [11], resolution reduction, real-time 3D compression [13, 14, 25], and multi-stream adaptation [15, 27]. However, multi-site tele-immersion poses unique challenges that have not been addressed. In particular, a simple yet efficient data dissemination model is needed to handle the coordinated delivery of large-volume data among multiple sites.

In this paper, we present a general publish-subscribe model to support multi-site 3DTI collaboration. Figure 3 shows the system architecture, in which the 3D cameras become the *publishers*, and the 3D displays become the *subscribers*. The subscription at each display is made by the local user as a preferred "field of view" (FOV) in the cyber-space, such that only the streams that are contributing to the FOV will be transmitted across the Internet. The FOV can be a subset of streams from other sites, or a particular rendering viewpoint of the cyber-space, for example. By leveraging user interest in a particular FOV, we can reduce the amount of required bandwidth without sacrificing much visual quality for the user. This is because the data that are not subscribed/delivered do not contribute to the user's field of view, thereby not noticeably affecting the visual quality.

Finally, a *rendezvous point (RP)* is introduced at each site to decouple the act of publishing from that of subscribing. An RP is basically a proxy server that is located at each site, to collect all streams from that site and disseminate them out to the network, as well as receive all streams intended for the participant at the local site. All RPs form an application-level overlay to disseminate only the data that are subscribed.

We find the static overlay construction problem among

Figure 2. With multi-site 3DTI, remote users can interact and collaborate in a "cyber-space".

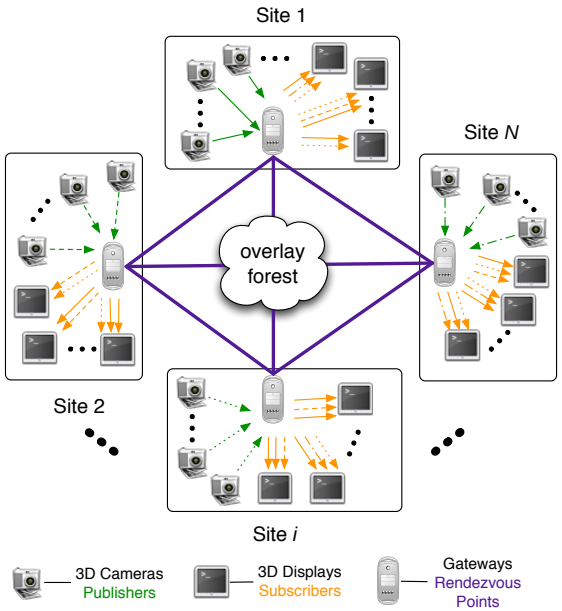


Figure 3. The Publish-Subscribe 3DTI System Model

all RPs to be an key challenge in the publish-subscribe model, due to several characteristics of multi-site 3DTI systems: (1) a dense graph: since a participant in 3DTI collaboration typically wants to see a large fraction of the other participants from a wide field of view, the overlay graph among all RPs has high density (i.e., the average in/out-degrees of all RP nodes are large) , and (2) multiple system constraints: each site has inbound/outbound bandwidth limits, and the end-to-end latency has to be small to guarantee interactivity.

In this paper, we tackle the static overlay construction problem with several tree-based heuristic algorithms and a randomized algorithm. Somewhat surprisingly we find that a simple randomized algorithm actually works well in the unique context. Furthermore, we observe that the randomized algorithm and the tree-based approaches are actually at two extreme ends of a general spectrum of algorithms. This leads us to study the whole spectrum of algorithms with a technique called "granularity analysis", which indeed confirms the advantages of the randomized algorithm. In light of these results, we propose optimization to further improve the randomized algorithm by exploiting semantic correlation among streams. We demonstrate that an improvement by a factor of five can be achieved.

Although this work was motivated by multi-stream/multi-site tele-immersion, the approaches can also be applied to other distributed multimedia application scenarios, such as multi-camera video conferencing,

distributed surveillance, and distance learning.

The remainder of the paper is organized as follows. Section 2 reviews the related work on 3DTI, publish-subscribe systems, and multicast overlay construction. Section 3 describes the design of the publish-subscribe model that leverages user interest. In Section 4, the overlay construction problem is discussed in great detail, together with a set of heuristic algorithms. Section 5 provides the experimental results, including the granularity analysis for evaluating the general spectrum of algorithms. Finally, we conclude and discuss future work in Section 6.

2 Background and Related Work

2.1 3D Tele-Immersion

3DTI has emerged as a new video medium for remote collaboration. However, the prior work on 3DTI mainly considered the multi-streaming challenge between only two sites. For example, Ott *et al.* [15] proposed a transport-level protocol for flow coordination, and Yang *et al.* [27] addressed the problem in the session layer with a multi-stream adaptation framework. Nevertheless, they did not handle multi-site interconnection and data delivery in the 3DTI environments.

Yang *et al.* [26] presented a general ViewCast model for distributed multimedia applications. The main idea of ViewCast was to decouple the high-level semantics of user customization from the low-level substrate for data dissemination. This model can be applied to multi-stream/multi-site 3DTI. However, the ViewCast model did not address the specific problem of distributing the large volume of data among multiple participating sites. Our work considers the practical challenges of the low-level data dissemination, and can serve as an underlying substrate for the ViewCast model.

2.2 Publish-Subscribe Systems

The publish-subscribe communication paradigm consists of three components: *publishers*, *subscribers*, and a *mediating infrastructure (brokers)*. The subscribers express interest in the data advertised by the publishers to the brokers. The publishers, unaware of who subscribe to what, simply deliver the data to the brokers. The brokers then match the subscribers' interests with the data produced by publishers, and deliver the matching content to the subscribers. The main advantage of the publish-subscribe paradigm is its capability to decouple the act of subscribing from the act of publishing with a mediating infrastructure. In the context of multi-site 3DTI systems, this leads to a conceptually simple networking model to handle the interconnection and data delivery among multiple participating sites.

There has been a large body of related work on publish-subscribe systems. The readers are referred to [8] for a good review. Our work differs in that it handles the delivery of live 3D video streams with large bandwidth consumption and stringent timing constraint. Furthermore, because a participant in 3DTI collaboration typically wants to see a large portion of other participants from a wide field of view, the delivery graph among all RPs has high density, which poses unique challenge for the static construction of the overlay structure for data dissemination.

2.3 Overlay Construction

We find that the static overlay construction problem in our publish-subscribe model is non-trivial. Related literature can be found on QoS routing for IP multicast [22] and overlay construction for application-level multicast (e.g., [4, 10, 16, 29]). However, the major concern there was the efficient or optimal construction of a single multicast tree on a certain constraint. We need to care about the coordinated construction of a large number of multicast trees on a particular set of nodes, subject to multiple constraints including bandwidth and latency. The forest construction needs to be carefully managed because the resources of the nodes are shared among all trees.

There has been also other work on application-level multicast (e.g., [3, 6, 18, 21, 30]) that aimed for scalability by using a structured overlay such as hierarchical clustering and DHTs. Since 3DTI session is usually small to medium sized, these structured techniques would incur unnecessary overhead and complexity in the control plane.

In the next section, we introduce the general publish-subscribe model for multi-site 3DTI systems. The problem of overlay construction among all RPs is presented in Section 4 in more detail.

3 Publish-Subscribe Model

3.1 Overview

We propose a general publish-subscribe model as a simple yet powerful data dissemination paradigm for multi-site 3DTI collaboration. The major advantages of the publish-subscribe model are twofold: (1) it decouples the cameras from the displays, such that the interconnection among multiple sites is greatly simplified; and (2) it leverages user interest as subscription to efficiently utilize the limited system resources, because only the data that are subscribed are transmitted.

Figure 3 shows the publish-subscribe model, where the 3D cameras that capture the local user in real time become

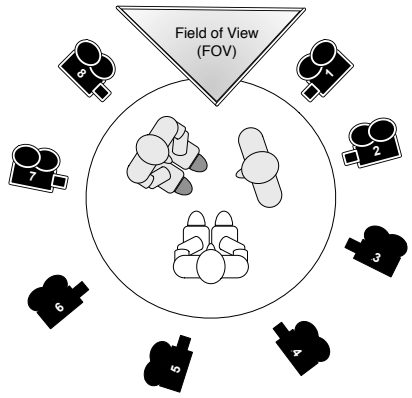


Figure 4. An example showing a preferred FOV in the 3DTI cyber-space and the contributing streams (those from cameras 1, 2, 7, 8).

the *publishers*, and the 3D displays that render the integrated cyber-space become the *subscribers*. Finally, a proxy server is placed at each site to act as the *rendezvous point (RP)*. Within a local site, the RP forms a *star* network to the cameras and the displays to handle publishing and subscribing, respectively. Among the RPs, an efficient application-level overlay is constructed to disseminate the subscribed streams across the Internet.

3.2 FOV-based Subscription

One major goal of the publish-subscribe model is to leverage the user interest as the subscription, in order to efficiently utilize the limited system resources. More specifically, the user configures a preferred *field of view* (FOV) in the cyber-space (Figure 2) as the subscription for each display, so that only the data that are contributing to the FOV will be transmitted. The FOV can be a subset of streams from other sites, or a particular rendering viewpoint of the cyber-space, for example. Figure 4 shows a FOV specified for a 3DTI cyber-space, and the streams produced from camera 1, 2, 7, 8 are the four most contributing streams to the selected FOV. By leveraging user interest in a particular FOV, we can reduce the amount of required bandwidth without sacrificing much visual quality for the user. This is because the data that are not subscribed/delivered do not contribute to the user's field of view, thereby not noticeably affecting the visual quality.

In order to allow the participants to specify a favored FOV for each display, a *subscription framework* is needed. We assume the design of the subscription framework is orthogonal to the publish-subscribe model, in order to decouple the high-level user customization semantics from the low-level stream dissemination substrate [26]. We only require the subscription framework to have two key functionalities: (1) allow the participants to specify a preferred FOV in a meaningful format (e.g., a rendering viewpoint of the cyber-space), and (2) convert the specified FOV to a concrete subset of streams that are contributing to the FOV. This subset of streams constitutes the actual *subscription requests* (i.e., which display subscribes to which stream), and will be fed into the overlay construction module (Section 4) as input. The insight of this design is that the proposed publish-subscribe model only serves as a substrate to handle the interconnection and data dissemination among multiple sites, while the high-level semantics of user interest in a FOV are left to the application designers, which results in better customization and flexibility[2].

The subscription requests from all displays are then collected by the local RP, and further aggregated to a centralized membership server[3]. Based on the global subscription workload (i.e., which site subscribes to which streams), the server dictates all RPs to organize into an application-level overlay network for data dissemination. After an RP collects the streams from other sites on the overlay, it distributes them to the local displays as requested.

It turns out that the static overlay construction problem in this context is non-trival (NP-complete). In this paper, we mainly tackle this static problem with a spectrum of heuristic algorithms, with the details presented in the next section.

4 Overlay Construction

As described above, within each site the RP forms a *star* network to the cameras and displays. However, the construction of the overlay among all RPs on the WAN is a key challenge in the publish-subscribe model.

4.1 Overview

Given all the subscription requests made via the subscription framework (Section 3.2), the main goal is to organize an overlay structure to disseminate the data among all RPs as requested. More specifically, each RP request only those streams that are subscribed by at least one of its local displays.

In the multi-stream/multi-site 3DTI environments, the overlay graph we are to construct is essentially a forest of multiple trees, where each tree is designated to disseminate

[2] As an example, ViewCast [26] can be used as a subscription framework, where for each display the user only needs to specify a preferred viewpoint of the cyber-space; a set of most correlated streams with respect to this viewpoint will be automatically selected to make the subscription requests.

[3] Since the 3DTI sessions are typically small to medium sized, we take the centralized approach for its simplicity.

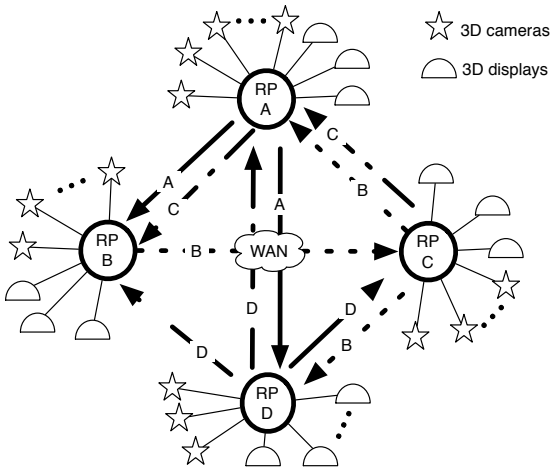


Figure 5. An example showing the publish-subscribe model of four sites. The label on each edge denotes the index of the source RP node for the transmitted stream.

one stream among the set of requesting RPs. The construction of such forest is complicated by several characteristics of multi-site 3DTI environments: (1) multiple constraints: each site has the inbound and outbound bandwidth limits, and the end-to-end delay between any pair of nodes has to be small in order to guarantee interactivity; (2) a dense graph: since the participant typically wants to see a large portion of other participants from a wide field of view, the overlay graph consisting of all RPs often has very high density (i.e., the average in/out-degrees of all nodes are large); hence, the construction of the forest needs to be carefully coordinated because the bandwidth resources are shared among all trees.

More concretely, we define a *multicast group*, $G(s)$, as the set of RP nodes that have requested the stream s. Note that we exclude the edge hosts (i.e., the cameras, the displays) from the overlay structure for the sake of simplicity. From hereafter, we use the terms nodes and RPs interchangeably.

For each multicast group $G(s)$, a multicast tree, T_s, needs to be constructed to disseminate the stream s from the source to all other nodes in $G(s)$. Figure 5 shows an overlay graph of four sites with four trees constructed among the RPs, {A, B, C, D}. For instance, RPs A, B, C, and D form a multicast group $G(s_B)$ that subscribe to the stream labeled "B" (i.e., the source is site B), and RP A, B, D form another multicast group $G(s_A)$ for stream labeled "A". The tree constructed to deliver stream s_B consists of the source node B, an intermediate node C, and two leaf nodes A, D.

In the next section, we consider the practical challenges of the overlay construction problem among all RPs , and provide a more formal definition.

Table 1. Notations

Notations	Comment
N	The total number of sites or RP nodes in a multi-site 3DTI session ($N \geq 3$)
H_i	$1 \leq i \leq N$, the site i that consists of the cameras, displays, and an RP node
RP_i	The RP node at site H_i
I_i, O_i	The inbound, outbound bandwidth limits at node $RP_i, respectively$
$d_{in}(RP_i)$, $d_{out}(RP_i)$	The actual inbound, outbound degree of node RP_i in the constructed overlay, respectively
s_j^q	The stream originating from site H_j with a local index q
$r_i(s_j^q)$	The subscription request specifying that RP_i requests the stream s_j^q
$G(s)$	The multicast group for stream s
T_s	The multicast tree constructed to disseminate stream s
F	The total number of multicast groups (or trees in the forest)
$u_{i\to j}, \hat{u}_{i\to j}$	$u_{i\to j}$ is the number of subscription requests made by node RP_i for node RP_j (i.e., $u_{i\to j}$ number of streams originating from site H_j are requested by RP_i), among which $\hat{u}_{i\to j}$ are rejected
X	The total request rejection ratio, defined in Equation 1
$m_i, \hat{m}_i$	m_i number of streams originate from site H_i and are subscribed by at least one other RP, among which $\hat{m}_i$ streams have not been disseminated out yet to any other RP.
$cost(RP_i, RP_j)_{T_s}$	The latency (or cost) between node RP_i and RP_j in tree T_s
B_{cost}	An upper bound for real-time latency to guarantee interactivity

4.2 Problem Formulation

Due to the huge demands of computing and networking resources in multi-site 3DTI collaboration, we have two

constraints and one optimization goal to satisfy in the overlay construction problem. To facilitate the discussion, a list of notations is shown in Table 1 as a reference.

- *Constraint I (bandwidth).* Each node has inbound (I_i) and outbound (O_i) bandwidth limits in the unit of number of streams (i.e., I_i, $O_i \in N$). A node should never receive data more than its inbound bandwidth limit (i.e., $d_{in}(RP_i) \leq I_i$ where $d_{in}(RP_i)$ is the actual in-degree of node RP_i in the overlay), nor be delegated to send data more than its outbound bandwidth constraint (i.e., $d_{out}(RP_i) \leq O_i$, where $d_{out}(RP_i)$ is the actual out-degree of RP_i).

- *Costraint II (latency).* In 3DTI, remote participants are rendered into the cyber-space in real time for interactive collaboration. Therefore, the expected end-to-end latency or cost between any pair of nodes, $cost(RP_i, RP_j)$ (for $1 \leq i, j \leq N$ and $i \neq j$), should not exceed a bound[4], B_{cost}, in order to guarantee interactivity.

- *Optimization Goal (request rejection ratio).* Due to the two stringent constraints listed above, we cannot guarantee that all subscription requests are satisfied. The metric we wish to minimize is the total rejection ratio of all requests in the system, denoted by X. Suppose the number of subscription requests made by node RP_i to RP_j is $u_{i\to j}$ (i.e., $u_{i\to j}$ number of streams originating from site H_j are subscribed by at least one display at site H_i), among which $\hat{u}_{i\to j}$ are rejected, we thus have

$$X = \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} \frac{\hat{u}_{i\to j}}{u_{i\to j}} \quad (1)$$

More specifically, the *forest construction problem* can be formulated as follows.

Forest Construction Problem. Given (1) a completely connected graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$, (2) an in-degree bound $I(v) \in N$, and an out-degree bound $O(v) \in N$, for each node $v \in \mathbb{V}$, (3) a cost $c(e) \in Z^+$ for each edge $e \in \mathbb{E}$, which denotes the latency, and (4) a set of subgraphs $\mathbb{G}_{subsets} = \{\mathbb{G}_i \mid \mathbb{G}_i = (\mathbb{V}_i, \mathbb{E}_i)$ where $\mathbb{G}_i \subseteq \mathbb{G}$ and $1 \leq i \leq |\mathbb{G}_{subsets}| = F\}$, each with a source node $S(\mathbb{G}_i) \in \mathbb{V}_i$, the goal is to find a *spanning forest*, $\mathbb{F} = \{\mathbb{T}_i \mid \mathbb{T}_i \subseteq \mathbb{G}_i$ for $1 \leq i \leq F\}$, with each tree $\mathbb{T}_i$ being a spanning tree that connects the source $S(\mathbb{G}_i)$ to a subset of the other nodes ($\mathbb{G}'_i \subseteq \mathbb{G}_i - S(\mathbb{G}_i)$), such that the total number of excluded nodes, $\sum_i |\mathbb{G}_i - \mathbb{G}'_i|$, is minimized, subject to the constraint that for all $v \in \mathbb{T}_i$ $(1 \leq i \leq F)$, $d_{in}(v) < I(v)$ and $d_{out}(v) < O(v)$, and $cost(v, S(\mathbb{G}_i))_{T_i} < B_{cost}$.

[4]As it is impossible to guarantee hard real-time bound in asynchronous network, we only attempt to satisfy an upper bound on expected latency from the source to the subscribers.

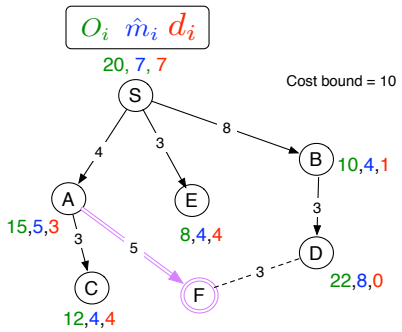


Figure 6. An example showing the basic algorithm to join a node into an existing tree.

Wang *et al.* [23] proved that the problem of finding a solution subject to two or more constraints in any combination in the multicast routing problem is NP-complete. In the next section, we study several heuristic algorithms to address the problem.

4.3 Heuristic Algorithms

As an overview, we discuss three tree-based algorithms and a randomized algorithm in this section. In all algorithms, the trees in the multicast forest are constructed incrementally, that is, within a multicast group $G(s)$, all requests for the stream s are processed sequentially in a randomized order. We formally define the *subscription request* as $r_i(s_j^q)$, specifying that RP_i requests to receive stream s_j^q which originates from site H_j with index q. Processing a request $r_i(s_j^q)$ thus means joining the node RP_i into the existing tree $T_{s_j^q}$. Section 4.3.1 describes a basic algorithm we propose for this purpose.

Furthermore, the order in which trees are constructed affects the overall optimization goal, due to the inter-dependencies among the trees. The inter-dependencies are caused by the shared limited resources of the nodes that are present in multiple trees. We propose three tree-based algorithms - LTF, STF, and MCTF - in Section 4.3.2, and a simple randomized algorithm in Section 4.3.3.

4.3.1 Basic Node Join Algorithm

We first describe a basic algorithm for processing one request $r_i(s_j^q)$, i.e., joining the node RP_i into the existing tree $T_{s_j^q}$. Recall that the 3DTI session involves a dense graph. Therefore, we seek for load balancing among all nodes such that no one would be particularly overloaded. The basic idea of this algorithm is thus to find a close-by node with

maximum available bandwidth left in the existing tree, to serve as the parent to the requesting node.

Before attempting to join the node RP_i into the tree $T_{s_j^q}$, the algorithm first checks the in-degree of RP_i. If $d_{in}(RP_i) < I_i$, it proceeds to the next step. Otherwise, it rejects the request because the inbound bandwidth limit is saturated.

After passing the inbound check, this basic algorithm looks for a parent node RP_k in the existing tree $T_{s_j^q}$ with free out-degree and the maximum *remaining forwarding capacity* (*rfc*) among all nodes in $T_{s_j^q}$, subject to the latency constraint that the cost from RP_i to the source of $T_{s_j^q}$ (i.e., RP_j) is smaller than a real-time bound (i.e., $cost(RP_i, RP_j)_{T_{s_j^q}} < B_{cost}$), if RP_i is connected to RP_k.

The rfc_i of node RP_i denotes the available portion of out-degree that can be used for forwarding streams. It is computed as $rfc_i = O_i - d_{out}(RP_i) - \hat{m}_i$ where $\hat{m}_i$ represents a reservation mechanism as follows. For each local stream RP_i has yet to send, (i.e., the stream is subscribed by at least one other node, but has not reached any node other than RP_i), a slot of out-degree is reserved to disseminate the stream out. Thus, $\hat{m}_i$ denotes the number of streams that (1) originate from node RP_i, (2) are subscribed by at least one other RP, but (3) have not yet been disseminated out to any other node in the existing forest. With this reservation mechanism, we minimize the probability that a whole tree cannot be constructed because the source node is saturated.

If no such eligible RP_k can be found in $T_{s_j^q}$, the request $r_i(s_j^q)$ is rejected. In this case, the tree is said to be *saturated*. The detailed algorithm is presented in pseudo code in the Appendix.

Figure 6 is an example where only one tree is shown for simplicity. F is the new node to join the existing tree of six nodes, {A, B, C, D, E, S} where S is the root. Among the nodes, E has no out-degree left to serve F (i.e., rfc=0), in which 4 is reserved for its out-streams ($\hat{m}_i$) and 4 is already taken in other trees ($d_{in}(RP_i)$). D has the largest *rfc* (22-8-0=14), but has a cost (8+3+3=14) exceeding the upper cost bound 10. A has the second largest *rfc* (15-5-3=7), and has a cost (4+5=9) smaller than the bound. Therefore, A becomes the parent to serve F. Again, this basic node join algorithm always seeks to achieve load balancing, which is essential in such a dense graph as a multi-site 3DTI session.

4.3.2 Tree-based Algorithms

Based on the basic node join algorithm, we propose three tree-based algorithms which differ by the order in which trees are constructed. We define the size of a tree T_s to be the number of nodes in the corresponding multicast group, i.e., $|G(s)|$. For each algorithm, the basic node join algorithm described in Section 4.3.1 is used to process a single request.

Largest Tree First (LTF) Algorithm. The intuition is to construct the largest tree first so that even if the last few trees cannot be constructed due to saturation, the number of rejected requests should be small because we are left with the smallest trees. More specifically, we first sort all multicast groups based on the size, and then construct the spanning trees one by one from the largest multicast group to the smallest one.

Smallest Tree First (STF) Algorithm. As a comparison to LTF, we also study this reversed algorithm which starts from the smallest multicast group, and ends with the largest one. Our hypothesis is that the rejection ratio of LTF should be smaller than that of STF.

Minimum Capacity Tree First (MCTF) Algorithm. This algorithm evaluates how difficult to construct a tree in terms of the aggregate *forwarding capacity* of a tree. The intuition is that the larger this value is, the easier it is to construct the tree. That is because new requests are easier to accommodate with a tree containing large aggregate forwarding capacity. A node RP_i's forwarding capacity can be computed as $O_i - m_i$, where m_i is the number of streams RP_i has to send out (i.e., the number of streams that originate from RP_i and are subscribed by at least one other RP). The forwarding capacity of a tree, T_s, is the aggregate forwarding capacity of all nodes in the multicast group $G(s)$. This algorithm sorts all multicast groups in the ascending order based on the aggregate forwarding capacity, and starts from the multicast group with the least capacity, to the one with the largest.

4.3.3 Randomized Algorithm (RJ)

Note that LTF, STF, and MCTF all seek to build the trees one by one, that is, only when it finishes processing *all* requests in one tree will it move on to construct the next one. In contrast, we propose a randomized algorithm, called "Random Join" (RJ), which simply randomizes all requests for the whole forest, with no prioritization on any tree. Again, the basic node join algorithm (Section 4.3.1) is used to process each request.

Somewhat surprisingly, our simulation in Section 5 finds that RJ generally outperforms the other tree-based algorithms. We will also show that the tree-based algorithms and the randomized algorithm are actually at two extreme ends of a more general spectrum of algorithms. This leads us to a more extensive study of the whole spectrum of algorithms (Section 5.3), which indeed confirms the advantages of RJ.

One of the reasons that the randomized algorithm works better is that every node in multi-site 3DTI collaboration is likely to be overloaded with subscription requests, because a participant typically wants to see a large portion of other

participants from a wide field of view. In tree-based algorithms, a node is much more likely to be congested in the first few constructed trees if it is the source, or a node near the source. This increases the probability of rejection in the construction of the latter trees because the node's total bandwidth is shared among different trees. In contrast, the randomized algorithm achieves good load balancing because it distributes the tasks of request processing among different trees randomly. The likelihood a single node is congested is thus minimized, and the rejection ratio decreased.

In light of these results, we next propose further optimization to the basic RJ algorithm by exploiting the semantic correlation among streams.

4.4 Exploiting Correlation

In 3DTI environments, the streams generated from one site have high semantic correlation among each other, because the cameras are often capturing the same scene, only from different angles (Figure 4). Therefore, we hope to exploit the inter-stream correlation to minimize the level of loss in times of saturation. As a motivating example, suppose a site A subscribes to four streams from site B (s_b^1, s_b^2, s_b^3, s_b^4) and one stream from site C (s_c^7). Then losing one stream from B is less critical than losing the single stream from C, since the former reduces the visual quality of a scene, while the latter loses a scene. Therefore, to minimize the level of loss for each participant, we *selectively* drop streams (i.e., reject requests) when the tree to join is saturated.

We describe a modified RJ algorithm, called CO-RJ, which exploits stream correlation. First, we introduce the concept of *criticality* for a node to lose a stream. Recall that $u_{i\to j}$ is the number of streams that node RP_i subscribes from node RP_j. The *criticality* for node RP_i to lose a stream s_j originating from RP_j is

$$Q_{i\to j} = \frac{1}{u_{i\to j}} \text{ for } 1 \le i, j \le N \text{ and } i \ne j \qquad (2)$$

In the previous example, the criticality for node A to lose any stream from site B is thus $\frac{1}{4}$, and that to lose s_c^7 is 1.

In CO-RJ, whenever a request is rejected due to tree saturation, the algorithm looks for a victim request with a smaller criticality value than the current request. If such a victim can be found, CO-RJ rejects the victim request, and satisfies the current request.

More specifically, when a request $r_i(s_j^p)$ (node RP_i requesting stream s_j^p) is rejected due to tree saturation, the algorithm checks the trees that have been constructed on the following conditions: (1) if there is a stream s_k^q ($k \ne j$) with $Q_{i\to k} < Q_{i\to j}$, and (2) RP_i is a leaf node in tree $T_{s_k^q}$ (or more simply T_k), and (3) the parent of RP_i in T_k - node RP_h - has already joined the tree $T_{s_j^p}$ (or more simply T_j), and (4) the cost between RP_i and the source for stream s_j^p (i.e., RP_j), if connecting RP_i to RP_h, is less than the real time bound (i.e., $cost(RP_i, RP_j)_{T_j} < B_{cost}$). If the four conditions are all satisfied, CO-RJ removes the edge $RP_h \to RP_i$ in T_k and add a new edge $RP_h \to RP_i$ in tree T_j. In other words, RP_i loses s_k^q instead of s_j^p. This operation is done with minimal cost, as RP_i was a leaf node in tree T_k, hence removing the old link would not cause relocation of any other nodes in T_k.

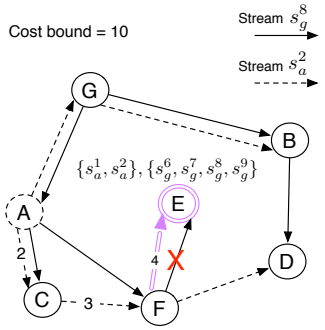


Figure 7. An example showing the CO-RJ algorithm. The label on the edge denotes the latency between the two nodes.

Figure 7 is an example showing two trees rooted at node A (for stream s_a^2) and G (for stream s_g^8), respectively. E has joined the tree for stream s_g^8 but wishes to receive stream s_a^2 too. E's subscription contains two streams from site A (s_a^1, s_a^2), and four streams from site G ($s_g^6, s_g^7, s_g^8, s_g^9$). Therefore, the criticality for E to lose a stream from A is $\frac{1}{2}$, and that from G is $\frac{1}{4}$, i.e., $Q_{E\to G} < Q_{E\to A}$. Assume the tree of s_a^2 is saturated, i.e., no eligible node can be found to serve E based on the bandwidth and delay constraints (Section 4.2.1). We have (1) $Q_{E\to G} < Q_{E\to A}$, (2) E is a leaf node in the original tree of s_g^8, which means removing it has no harm for the other nodes, (3) node F, which is the parent of E in tree s_g^8, actually has the stream s_a^2, and (4) if connecting E to F in the tree of s_a^2, the cost (2+3+4=9) would be smaller than the bound. Since all four conditions are satisfied, CO-RJ will remove the link F→E in the tree of s_g^8, and add the link F→E in the tree of s_a^2 as shown in Figure 7. In other words, F serves E with the new stream s_a^2 instead of s_g^8 although F itself is saturated. Because for E, s_g^8 is less critical to lose than s_a^2.

5 Evaluation

5.1 Simulation Setup

Topology. We use the real Internet topology (i.e., Mapnet [1]) to evaluate the algorithms. We randomly select 3-10

nodes in the experiments. The costs of edges are computed based on the geographical distances between the nodes.

Node Resource Distribution. We configure the experiment parameters close to real-life settings. According to the measurement by our implemented 3DTI system [19, 28], the available bandwidth of tele-immersive sites on Internet2 could vary between 40Mbps and 150Mbps, and a 3D video stream after using a series of reduction techniques (e.g., background subtraction [11], resolution reduction, real-time 3D compression [25, 14]) is approximately 5-10Mbps. We evaluate two types of node capacity distribution: (1) *uniform*: a capacity of $O_i = I_i = 20 \pm \epsilon$ where $1 \leq i \leq N$ and ϵ is uniformly distributed between 0 and 5. The number of streams each site has to send is 20. (2) *heterogeneous*: fifty percent of the nodes have large capacity (30), twenty-five percent have medium capacity (20) and the other twenty-five percent have small capacity (10). The number of streams each site has to send is uniformly distributed between 10 and 30.

Subscription Workloads. We mainly evaluate two types of subscription workloads: (1) *Zipf-distributed*: it has been shown that the stream popularity in multimedia applications follows a Zipf-like distribution [5, 7, 20]. We find this to be intuitively true in 3DTI environments, as the front cameras that capture people's faces are likely to be subscribed by most sites. (2) *random*: the randomized workload is to account for the possibility that the streams have more or less similar popularity in some 3DTI applications, such as surveillance and group collaboration. For both Zipf-distributed and random workloads, two hundred samples are generated to enumerate the possible subscriptions (i.e., which streams are subscribed by which sites).

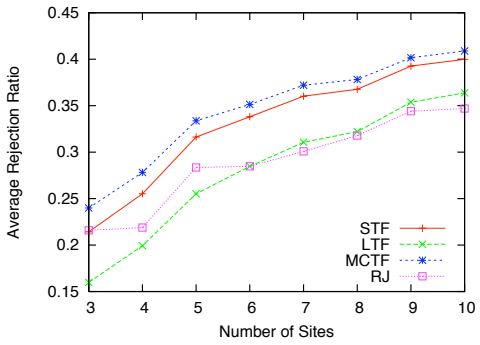
5.2 Rejection Ratio

Figure 8 shows the average[5] rejection ratios (defined in Section 4.2) achieved by the tree-based algorithms and the basic randomized algorithm, under different node resource distribution and subscription workloads.

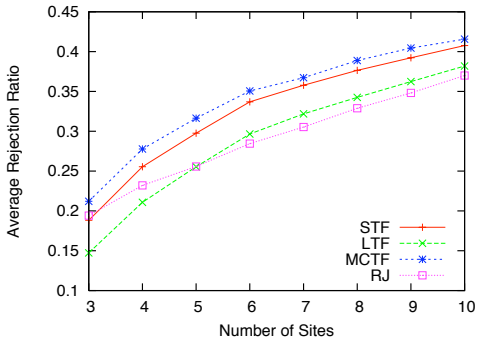
First, we notice the general trend is that the rejection ratio is increasing with the number of sites. This is because the total subscription workload grows much faster than the total available resources to serve the subscription requests. The resource per node is almost constant, whereas the subscription load grows with the total number of available streams.

Second, the data support our hypothesis that the LTF algorithm should perform better than STF. For example, with heterogeneous nodes under random workload (Figure 8(c)), LTF is about 25% better than STF. The rationale is that even

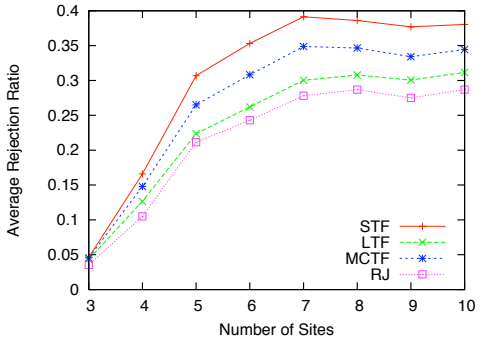
[5]We take the average rejection ratio across the two hundred random samples of subscription workloads.



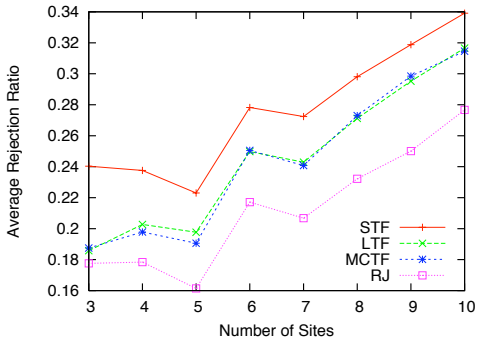
(a) Zipf workload, heterogeneous nodes



(b) Zipf workload, uniform nodes



(c) Random workload, heterogeneous nodes



(d) Random workload, uniform nodes

Figure 8. Average rejection ratios of different algorithms.

if the last few trees cannot be constructed because of saturation, the number of rejected requests should be small because we are left with the smallest trees.

Third, as mentioned before, somewhat surprisingly RJ generally achieves the lowest rejection ratio in different experimental settings. For example, with uniform nodes under random workload (Figure 8(d)), RJ is about 26.7% better than STF, while 16.7% better than LTF and MCTF. Although LTF sometimes obtains close performance to RJ (Figure 8(a) and 8(b)), it is computationally more expensive, because tree-based algorithms requires sorting of all multicast groups, while RJ just randomly picks requests to serve. Therefore, RJ turns out to be the simplest but the most favorable solution in the unique problem context.

5.3 Granularity Analysis

As mentioned earlier, we observe that the RJ algorithm and the tree-based algorithms (LTF, STF, MCTF) are actually at two extreme ends of a more general spectrum of algorithms. We thus perform more extensive experiments to gain insight into the difference between the two types of approaches.

Recall that the tree-based algorithms (i.e., LTF, STF, MCTF) all construct the trees one by one, while RJ attempts to construct the *forest* at once. We thus define the number of trees an algorithm attempts to construct at once as the *granularity*, g ($1 \leq g \leq F$ and $g \in N$, where F is the total number of multicast groups, or trees to construct). As two extreme cases, the granularity of all aforementioned tree-based algorithms is 1, while that of the randomized algorithm is F.

To better understand the impact of granularity on rejection ratio, we perform experiments by incrementally increasing the granularity value. A modified LTF algorithm, called Gran-LTF, is used in this experiment as it is the best tree-based algorithm among the three.

Instead of constructing the trees one by one as in the original LTF algorithm, Gran-LTF first sorts all multicast groups in a descending order based on the size of the groups. It then picks the first g (number of) multicast groups for spanning tree construction. Note that within the set of g multicast groups (thus g trees), the requests are processed randomly using the basic node join algorithm (Section 4.3.1). Only after finishing processing all requests in the g multicast groups, the algorithm proceeds to the pick the next g trees to construct, and so forth.

Figure 9 shows the result with ten uniform nodes under random workload. Note that when $g = F$, Gran-LTF becomes RJ. We observe that generally the larger the granularity, the lower the rejection ratio. Although there is a small fluctuation region in the end (where granularity is large), the basic RJ algorithm is computationally simpler than others.

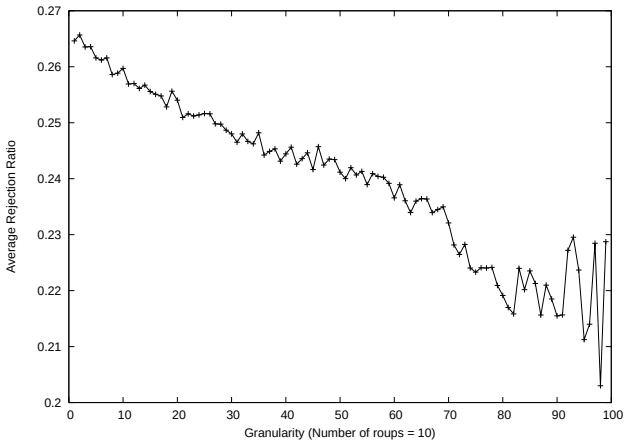


Figure 9. Impact of granularity on rejection ratio (N=10)

The graphs for other experimental settings look similar to Figure 9 when N grows from 3 to 10, which we omit due to the space limit.

5.4 Load Balancing

As mentioned earlier, load balancing is important to achieve low rejection ratio in the bandwidth-demanding and computation-intensive 3DTI environments. Figure 10 shows the average out-degree utilization achieved by RJ with uniform nodes under random workload. First, we notice that the average out-degree utilization is very high - mostly close to 100%. Second, the standard deviation of the out-degree utilization across all nodes is small ($< 3\%$). This shows that our algorithms achieve good load balancing because each node takes approximately the same amount of bandwidth to disseminate streams. Third, about 25% of each node's out-degree is dedicated for forwarding the streams originating from other nodes. This indicates the strength of our multicast-based protocol that substantially reduces the burden of each source node compared to the conventional unicast "all-to-all" solution.

5.5 Correlation

Finally, we compare the CO-RJ algorithm (Section 4.4) with the original RJ algorithm (Section 4.3.3). Figure 11 shows the result with heterogeneous nodes under Zipf-distributed workload. In order to account for stream correlation, the definition of rejection ratio is modified as follows.

$$X' = \sum_{i=1}^{N} (\sum_{j=1}^{N} \frac{\hat{u}_{i \to j}}{u_{i \to j}^2}) \cdot u_{i \to x} \qquad (3)$$

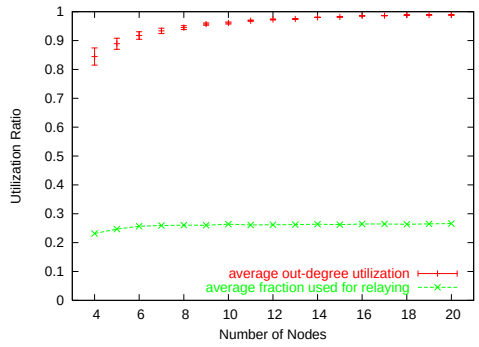


Figure 10. Average out-degree utilization

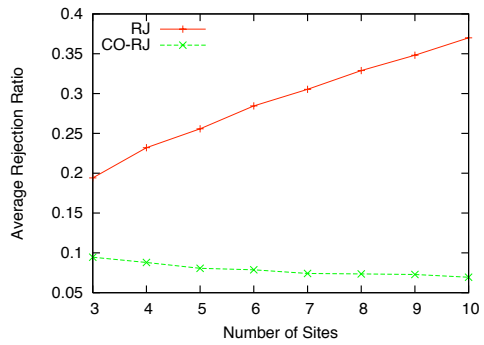


Figure 11. Average rejection ratio with correlation taken into account

where $u_{i \to x} = min(u_{i \to j})$ for $1 \leq j \leq N$. Figure 11 shows that CO-RJ's rejection ratio decreases as the number of sites grows, while RJ performs worse. When $N = 10$, CO-RJ is a factor of 5 better than RJ, which demonstrates the strength of the optimization based on stream correlation.

6 Conclusion and Future Work

In this paper, we considered the practical challenges in supporting multi-site 3DTI collaboration. We proposed a general publish-subscribe model to handle the interconnection and data dissemination in the multi-stream/multi-site environments. The publish-subscribe model leveraged user interest in a field of view to efficiently utilize the limited network resources.

We found the key challenge in the publish-subscribe model was the static construction of an efficient overlay, to deliver the live video streams among multiple 3DTI sites subject to several system constraints. We explored a spectrum of heuristic algorithms to address the challenge. We found that a simple randomized algorithm worked well in this problem context. We hence proposed further optimization to the basic randomized algorithm by exploiting stream correlation. The experimental results demonstrated that the optimization mechanism achieved significant improvement over the basic algorithm.

As future work, we hope to collect a large amount of user traces for subscription workloads in 3DTI ennvironments, and perform more extensive experiments to evaluate the heuristic algorithms. Moreover, we are in the processing of applying the publish-subscribe model to work with ViewCast [26], which may lead to experiments of larger scales with real deployment on the Internet.

References

[1] Mapnet. *http://www.caida.org/tools/visualization/mapnet/Data/*.

[2] H. Baker, N. Bhatti, D. Tanguay, I. Sobel, D. Gelb, M. Goss, W. Culbertson, and T. Malzbender. The coliseum immersive teleconferencing system. *ACM Trans. on Multimedia Computing, Communications, and Applications*, 2005.

[3] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. In *ACM SIGCOMM*, 2002.

[4] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller. Construction of an efficient overlay multicast infrastructure for real-time applications. In *Proceedings of INFOCOM*, San Francisco, CA, 2003.

[5] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and zipf-like distributions: Evidence and implications. In *Proc. of IEEE INFOCOM 1999*, March 1999.

[6] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. Splitstream: High-bandwidth multicast in cooperative environments. In *19th ACM Symposium on Operating Systems Principles (SOSP)*, 2003.

[7] M. Chesire, A. Wolman, G. M. Voelker, and H. M. Levy. Measurement and analysis of a streaming media workload. pages 1–12.

[8] P. Eugster, P. Felber, R. Guerraoui, and A. Kermarrec. The many faces of publish/subscribe. In *ACM Computing Surveys*, pages 114–131, 2003.

[9] K. E. Finn, A. Sellen, S. Wilbur, K. Finn, A. J. Sellen, and S. B. Wilbur. *Video-mediated Communication*. Lawrence Erlbaum Associates, 1997.

[10] Y. hua Chu, S. G. Rao, S. Seshan, and H. Zhang. A case for end-system multicast. *IEEE Journal on Selected Areas in Communications special issue on Network Support for Multicast Communications*, 20(8), Oct. 2002.

[11] S. Jung and R. Bajcsy. A framework for constructing real-time immersive environments for training physical activities. In *Journal of Multimedia*, volume 1, pages 9–17, 2006.

[12] S. Jung and R. Bajcsy. Learning physical activities in immersive virtual environments. In *Proceedings of the 4th IEEE Int. Conference on Computer Vision Systems*, New York City, January 5-7th, 2006.

[13] S. Kum, K. Mayer-Patel, and H. Fuchs. Real-time compression for dynamic 3d environments. In *Proceedings of the eleventh ACM international conference on Multimedia*, 2003.

[14] J. Lien, G. Kurillo, and R. Bajcsy. Skeleton-based data compression for multi-camera tele-immersion system. In *Proc.*

of International Symposium on Visual Computing (ISVC 2007), Lake Tahoe, CA, 2007.

[15] D. E. Ott and K. Mayer-Patel. An open architecture for transport-level protocol coordination in distributed multimedia applications. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 3, 2007.

[16] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel. Almi: An application level multicast infrastructure. In *Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems (USITS)*, 2001.

[17] R. Raskar, G. Welch, M. Cutts, A. Lake, L. Stesin, and H. Fuchs. The office of the future: A unified approach to image-based modeling and spatially immersive displays. In *Proc. of ACM SIGGRAPH 98*, Orlando, FL, 1998.

[18] A. Rowstron, A. Kermarrec, M. Castro, and P. Druschel. Scribe: The design of a large-scale event-notification infrastructure. In *Proceedings of the 3rd International Workshop on Network Group Communications*, pages 30–43, November 2001.

[19] R. Shepherd, W. Wu, Z. Yang, K. Nahrsted, L. Wymore, G. Kurillo, R. Bajcsy, and K. Mezur. Teeve: New digital options for collaborative dance in geographically distributed tele-immersive spaces. In *ACM Multimedia Demo Session*, 2007.

[20] K. Sripanidkulchai, B. Maggs, and H. Zhang. An analysis of live streaming workloads on the internet. In *4th ACM SIGCOMM conference on Internet measurement*, pages 41–54, New York, NY, USA, 2004. ACM Press.

[21] D. Tran, K. Hua, and T. Do. Zigzag: An efficient peer-to-peer scheme for media streaming. In *IEEE INFOCOM*, 2003.

[22] B. Wang and J. Hou. Multicast routing and its QoS extension: problems, algorithms, and protocols. *IEEE Network*, 14, Jan. 2000.

[23] Z. Wang and J. Crowcroft. Qos routing for supporting resource reservation. In *IEEE Journal on Selected areas in Communications*, Sep. 1996.

[24] G. Welch, D. H. Sonnenwald, K. Mayer-Patel, R. Yang, A. State, H. Towles, B. Cairns, and H. Fuchs. Remote 3d medical consultation. In *Proc. of the 2nd International Conference on Broadband Networks*, 2005.

[25] Z. Yang, Y. Cui, Z. Anwar, R. Bocchino, N. Kiyanclar, K. Nahrstedt, R. H. Campbell, and W. Yurcik. Real-time 3d video compression for tele-immersive environments. In *Proc. of SPIE/ACM Multimedia Computing and Networking (MMCN'06)*, San Jose, CA, 2006.

[26] Z. Yang, W. Wu, K. Nahrstedt, G. Kurillo, and R. Bajcsy. Viewcast: View dissemination and management for multi-party 3d tele-immersive environments. In *Proc. of Proc. of 15th ACM International Conference on Multimedia*, 2007.

[27] Z. Yang, B. Yu, K. Nahrstedt, and R. Bajcsy. A multi-stream adaptation framework for bandwidth management in 3d tele-immersion. In *Proc. of the 16th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'06)*, Newport, Rhode Island, May 22-23 2006.

[28] Z. Yang, B. Yu, W. Wu, R. Diankov, and R. Bajcsy. A study of collaborative dancing in tele-immersive environment. In *Proc. of the 8th IEEE International Symposium on Multimedia (ISM'06)*, San Diego, CA, 2006.

[29] B. Zhang, S. Jamin, and L. Zhang. Host multicast: A framework for delivering multicast to end users. In *Proceedings of IEEE Infocom*, 2002.

[30] S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. H. Katz, and J. D. Kubiatowicz. Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. In *Proceedings of NOSSDAV*, 2001.

A Basic Node Join Algorithm

Algorithm 1 $nodejoin$: Join node RP_i into the existing tree $T_{s_j^p}$. Return True if successfully joined, False if rejected

if $d_{in}(RP_i) \geq I_i$ **then**
  return False
end if
$max \leftarrow 0$
$candidate \leftarrow -1$
for all nodes RP_k in $T_{s_j^p}$ where $k = 1$ to $|T_{s_j^p}|$ **do**
  if RP_k has stream s_j^p and $d_{out}(RP_k) < O_k$ and $cost(RP_i, RP_j)_{T_{s_j^p}} < B_{cost}$ **then**
    if $k = j$ and ((reserved[k][p]=0) or (reserved[k][p] = 1 and $O_k - \hat{m} > max$)) **then**
      $candidate \leftarrow k$
      $reserved[k][p] \leftarrow 1$
    else
      if $k \neq j$ and $O_k - \hat{m}_k - d_{out}(RP_k) > max$ **then**
        $candidate \leftarrow k$
        $max \leftarrow O_k - \hat{m}_k - d_{out}(RP_k)$
      end if
    end if
  end if
end for
if $candidate \neq -1$ **then**
  add link $RP_{candidate} \rightarrow RP_i$ in $T_{s_j^p}$
  return True
else
  return False
end if