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 to use 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 present 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 (3DTI) 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 distance learning of physical activities, collaborative art performance, firefighter training, and medical consultation [2, 9, 17, 22].

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 ("cyber-space") can be constructed that immerses the participants from all sites, as illustrated in Figure 1. The shared visual context enables remote users to interact or collaborate "virtually" in the cyber-space.

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, 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 limit and the timing requirement for interactivity.

Therefore, the "all-to-all" data distribution scheme, adopted by existing 2D video-conferencing and 3DTI systems ([2, 6]), has to be abandoned as the scale of the 3DTI system grows. As a concrete example, the 3DTI system described by Yang *et al.* [22] involved two sites thousands of miles apart, 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, resolution reduction, real-time 3D compression [10, 11, 19], and multi-stream adaptation [12, 21]. 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 show how the general publish-subscribe model can be used to support multi-site 3DTI collaboration, which greatly simplifies the interconnection among multiple sites and hence increases the scalability of the system.

*This research is supported by National Science Foundation (NSF) under grants CNS-0448246, CMS-0427089, NSF SCI 05-49242 and NSF CNS 05-20182. The presented views are those of authors and do not represent the position of NSF.

For the subscription semantics, we leverage the user interest in a particular "field of view" (FOV) for the cyber-space, such that only a subset of streams that are contributing to the FOV will be transmitted across the Internet. The key insight is that 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.

We find the static overlay construction problem among all sites to be a key challenge (NP-complete) in the publish-subscribe model.We tackle the problem with several tree-based heuristic algorithms and a randomized algorithm. Somewhat surprisingly we find that the 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, which indeed confirms the advantages of the randomized algorithm. In light of these results, we propose optimizations 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. We review the related work in Section 2, and introduce the 3DTI publish-subscribe model in Section 3. Section 4 is dedicated to the overlay construction problem and the spectrum of heuristic algorithms. We provide the experimental results in Section 5. Finally, we conclude and discuss future work in Section 6.

Figure 1. Multi-site 3DTI collaboration in a cyber-space

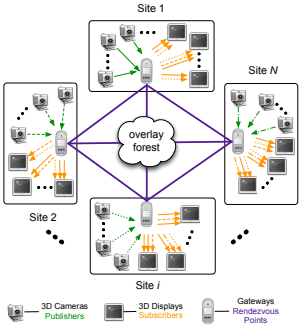


Figure 2. Publish-Subscribe 3DTI System

2 Background and Related Work

The prior work on 3DTI mainly considered the multi-streaming challenge between only two sites. For example, Ott *et al.* [12] proposed a transport-level protocol for flow coordination, and Yang *et al.* [21] addressed the problem in the session layer with a multi-stream adaptation framework. Nevertheless, they did not handle the interconnection and data delivery for multi-site collaboration.

Yang *et al.* [20] presented a general ViewCast model for distributed multimedia applications, which can be applied to multi-site 3DTI. However, it did not address the 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.

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 [15] and overlay construction for application-level multicast (e.g., [3, 7, 23]). 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., [4, 13, 14]) 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.

3 Publish-Subscribe Model

We propose to use the general publish-subscribe model as a simple yet powerful data dissemination paradigm for

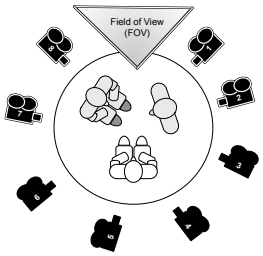


Figure 3. FOV and the contributing streams

multi-site 3DTI collaboration. The publish-subscribe communication paradigm consists of three components: *publishers*, *subscribers*, and a *mediating infrastructure, i.e., rendezvous points (RP)*. The subscribers express interest in the data advertised by the publishers to the RPs. The publishers, unaware of who subscribe to what, simply deliver the data to the RPs. The RPs then match the subscribers' interests with the data produced by publishers, and deliver the matching content to the subscribers.

Figure 2 shows the publish-subscribe 3DTI system, 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. Figure 3 shows an example of FOV and its contributing streams (those from camera 1, 2, 7, 8).

Note that we assume the design of a subscription framework for users to specify FOV is *orthogonal* to our problem. This is to decouple the high-level user customization semantics from the low-level stream dissemination substrate [20]. We only require the subscription framework to have the capability of converting the user-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 displays subscribes to which streams)*, and will be fed into the overlay construction module (Section 4) as input. ViewCast [20] is an example of such subscription framework that selects a set of most correlated streams with respect to a specified viewpoint.

Finally, a proxy server is placed at each site to act as the *rendezvous point (RP)*. RP's main role is 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. RPs also collect the subscription requests from all its local displays, and further aggregate them to a centralized membership server[1]. Based on the global subscription workload, 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.

The major advantages of the publish-subscribe model are twofold: (1) it decouples the cameras from the displays, so that the interconnection among multiple sites is greatly simplified; and (2) it leverages user interest in FOVs as subscription semantics, so that the limited system resources can be efficiently utilized.

4 Overlay Construction

As shown in Figure 2, within each site the RP forms a *star* topology 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.

Given all the subscription requests, the main goal is to organize an overlay structure to disseminate the data among all RPs as requested. In the multi-stream/multi-site 3DTI environments, the overlay graph we are to construct is essentially a *forest* of multiple trees, with each tree designated to disseminate a stream among the set of requesting RPs. We define the *multicast group*, $G(s)$, as the set of RP nodes that have requested (i.e., one of its displays subscribed to) the stream s. 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.

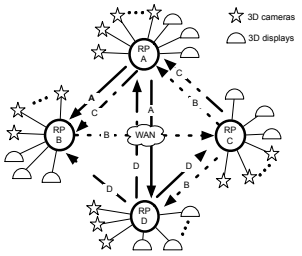


Figure 4. Overlay structure

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. Note that each tree T_s only includes the RPs in $G(s)$. The RPs that are not requesting the stream s are not used in T_s because they are already highly loaded with their streams. Figure 4 shows an overlay graph of four sites with four trees constructed among the RP nodes $\{$A, B, C, D$\}$. The label on each edge denotes the index of the source RP for the transmitted stream. For instance, RPs A, B, C, and D form a multicast group $G(s_B)$ that subscribe to the stream from B, and RPs A, B, D form another group $G(s_A)$ for stream from A. Note that this is only a simplified

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

example, as each node is the source of a single tree. In reality, each site often has tens of streams to disseminate, and hence it acts as a source for that many trees.

The construction of such a forest is complicated by several characteristics of multi-site 3DTI environments: (1) multiple system constraints: each site has 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.

4.1 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.

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, $O_i \in N$), which can be dynamically measured by existing probing tools like Pathload [8]. 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[2], 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 $X = \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} \frac{\hat{u}_{i\to j}}{u_{i\to j}}$.

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 fraction of excluded nodes, $\sum_i (|\mathbb{G}_i - \mathbb{G}'_i|/\mathbb{G}_i)$, 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 $cost(v, S(\mathbb{G}_i))_{T_i} < B_{cost}$.

Wang *et al.* [16] 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. We study several heuristic algorithms to address the problem.

4.2 Heuristic Algorithms

We will discuss three tree-based algorithms and a randomized algorithm. In all cases, 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. Section 4.2.1 describes a basic algorithm for this purpose. 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 describe three tree-based algorithms - LTF, STF, and MCTF - in Section 4.2.2, and a simple randomized algorithm in Section 4.2.3.

4.2.1 Basic Node Join Algorithm

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. The basic node join algorithm is used to process a request $r_i(s_j^q)$, i.e., joining the node RP_i into the existing tree $T_{s_j^q}$. Since the 3DTI session involves a dense graph, we desire *load balancing* among all nodes such that no one would be particularly overloaded. The basic idea is thus to find a "close-by" node (for the concern of latency) with the *maximum* available bandwidth left in the existing tree, to serve as the parent to the requesting node. Several metrics can be used to find the "close-by" node, such as network coordinates, round-trip time measurement, and geographical distances, and we choose to use the last one in our experiments.

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

[2]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 destinations.

$d_{in}(RP_i) < I_i$, it proceeds to the next step. Otherwise, it rejects the request because the inbound bandwidth limit is saturated. After passing the inbound check, the algorithm looks for a parent node RP_k in the existing tree $T_{s_j^q}$ with available 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) would be smaller than a real-time bound, if RP_i were 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 $\hat{m}_i$ 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. This reservation mechanism ensures that 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 pseudo code of the algorithm can be found in the authors' technical report [18].

Figure 5(a) 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 seeks to achieve load balancing, which is essential in such a dense graph as a multi-site 3DTI session.

4.2.2 Tree-based Algorithms

We now describe three tree-based algorithms which differ in the order of tree construction. For each algorithm, the basic node join algorithm (Section 4.2.1) 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 rejection ratio should be small because we are left with the smallest trees. The size of a tree T_s is intuitively the number of nodes in the corresponding multicast group, i.e., $|G(s)|$. 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 considers the difficulty of tree construction in terms of the *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 is $O_i - m_i$, 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 sum of the 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.

4.2.3 Randomized Algorithm (RJ)

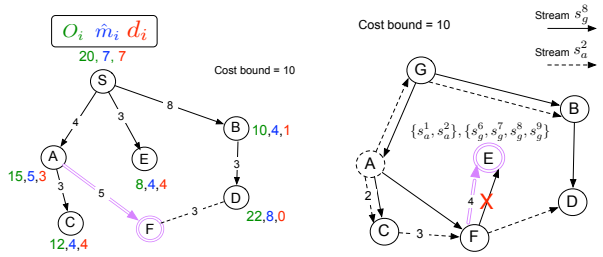
LTF, STF, and MCTF all seek to build the trees one by one, that is, only when it finishes processing *all* requests in one tree will it move on to construct the next one. In contrast, we propose a randomized algorithm, called "Random Join" (RJ), which randomizes all requests for the whole forest, with no prioritization on any tree. Again, the basic node join algorithm (Section 4.2.1) is used to process each request.

Somewhat surprisingly, our simulation in Section 5 finds that RJ generally outperforms the other tree-based algorithms. 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.

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

4.3 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 3). We exploit the inter-stream correlation to minimize the level of loss in times of saturation. As a motivating example, suppose a site A subscribes



(a) Basic node join algorithm (b) CO-RJ algorithm

Figure 5. Examples of algorithms

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 $Q_{i\to j} = \frac{1}{u_{i\to j}}$ for $1 \leq i, j \leq N$ and $i \neq j$. 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 four conditions: (1) if there is a stream s_k^q ($k \neq j$) 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 5(b) is an example showing two trees rooted at node A (for stream s_a^2) and G (for stream s_g^8), respectively. The label on the edge denotes the latency between the two nodes. 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 4.1). 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, (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 5(b). In other words, F serves E with the new stream s_a^2 instead of s_g^8 although F itself is saturated.

5 Evaluation

5.1 Simulation Setup

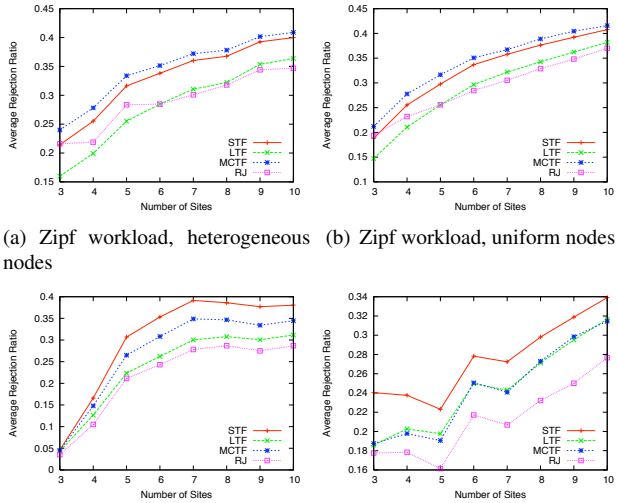
Topology. We use the real Internet topology (i.e., Mapnet [1]) to evaluate the algorithms. We randomly select 3-10 nodes in the experiments. The costs of edges are computed based on the geographical distances between the nodes.

Node Resource Distribution. We configure the experiment parameters close to real-life settings. According to the measurement by our implemented 3DTI system [22], 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, resolution reduction, real-time 3D compression [19, 11]) 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 chosen uniformly 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]. 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



(a) Zipf workload, heterogeneous nodes (b) Zipf workload, uniform nodes

(c) Random workload, heterogeneous nodes (d) Random workload, uniform nodes

Figure 6. Rejection ratios

Figure 6 shows the average rejection ratios (defined in Section 4.1) 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 6(c)), LTF is about 25% better than STF. The rationale is that even 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 6(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 6(a) and 6(b)), it is computationally more expensive, because tree-based algorithms require 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

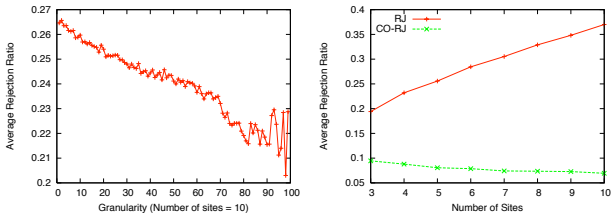
We observe that the RJ algorithm and the tree-based algorithms (LTF, STF, MCTF) are actually at two extreme ends of a more general spectrum of algorithms. We define the number of trees an algorithm attempts to construct at once as the *granularity*, g ($1 \leq g \leq F$ and $g \in N$, where F is the 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. We perform experiments by incrementing the granularity value.

A modified LTF algorithm, called Gran-LTF, is used in this experiment as it is the best-performing 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. Within the set of g multicast groups (thus g trees), the requests are processed randomly using the basic node join algorithm (Section 4.2.1). 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 7(a) 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. The graphs for other experimental settings look similar to Figure 7(a) when N grows from 3 to 10, which we omit due to the space limit.

5.4 Correlation

Finally, we compare the CO-RJ algorithm (Section 4.3) with the original RJ algorithm (Section 4.2.3). Figure 7(b) 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: $X' = \sum_{i=1}^{N}(\sum_{j=1}^{N} \frac{\hat{u}_{i\to j}}{u^2_{i\to j}}) \cdot u_{i\to x}$, where $u_{i\to x} = min(u_{i\to j})$ for $1 \leq j \leq N$. Figure 7(b) 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.



(a) Impact of granularity (b) Impact of correlation

Figure 7. Granularity and correlation

6 Conclusion and Future Work

In this paper, we considered the practical challenges in supporting multi-site 3DTI collaboration. We proposed to use a general publish-subscribe model to handle the interconnection and data dissemination in the multi-stream/multi-site environments. We studied the key challenge of static overlay construction by a spectrum of heuristic algorithms.

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 [20], which may lead to experiments of larger scales with real deployment on the Internet.

7 Acknowledgment

We thank our collaborators Gregorij Kurillo, Ruzena Bajcsy, and Peter Bajcsy for their great support. We also thank the anonymous reviewers for their helpful comments. Wanmin Wu was awarded the Yahoo! Key Technical Challenge grant to present this paper at the conference.

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. Understanding performance in coliseum, an immersive videoconferencing system. *ACM TOMCCAP*, 1(2):190–210, 2005.

[3] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller. Construction of an efficient overlay multicast infrastructure for real-time applications. In *Proc. of IEEE INFOCOM*, volume 2, pages 1521–1531, 2003.

[4] 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.

[5] 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.

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

[7] Y. hua Chu, S. G. Rao, S. Seshan, and H. Zhang. A case for end-system multicast. *IEEE J-SAC*, 20(8):1456–1471, 2002.

[8] M. Jain and C. Dovrolis. End-to-end available bandwidth: Measurement methodology, dynamics, and relation with tcp throughput. In *Proc. of ACM SIGCOMM*, pages 295–308, 2002.

[9] 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.

[10] S. Kum, K. Mayer-Patel, and H. Fuchs. Real-time compression for dynamic 3d environments. In *Proc. of ACM Multimedia*, pages 185–194, 2003.

[11] J. Lien, G. Kurillo, and R. Bajcsy. Skeleton-based data compression for multi-camera tele-immersion system. In *Proc. of ISVC*, pages 714–723, Lake Tahoe, CA, 2007.

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

[13] A. Rowstron, A. Kermarrec, M. Castro, and P. Druschel. Scribe: The design of a large-scale event-notification infrastructure. In *Proc. of NGC*, pages 30–43, 2001.

[14] D. Tran, K. Hua, and T. Do. Zigzag: An efficient peer-to-peer scheme for media streaming. In *Proc. of IEEE INFOCOM*, volume 2, pages 1283–1292, 2003.

[15] B. Wang and J. Hou. Multicast routing and its QoS extension: problems, algorithms, and protocols. *IEEE Network*, 14(1):22–36, 2000.

[16] Z. Wang and J. Crowcroft. Qos routing for supporting resource reservation. In *IEEE J-SAC*, volume 14, pages 1228–1234, 1996.

[17] 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 BROADNETS*, volume 2, pages 1026–1033, 2005.

[18] W. Wu, Z. Yang, I. Gupta, and K. Nahrstedt. Towards multi-site collaboration in 3d tele-immersive environments. Technical Report UILU-ENG-2008-1726, University of Illinois at Urbana-Champaign, 2008.

[19] 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 MMCN*, San Jose, CA, 2006.

[20] 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 ACM Multimedia*, pages 882–891, 2007.

[21] Z. Yang, B. Yu, K. Nahrstedt, and R. Bajcsy. A multi-stream adaptation framework for bandwidth management in 3d tele-immersion. In *Proc. of NOSSDAV*, Newport, Rhode Island, May 22-23 2006.

[22] Z. Yang, B. Yu, W. Wu, R. Diankov, and R. Bajcsy. A study of collaborative dancing in tele-immersive environment. In *Proc. of ISM*, pages 177–184, San Diego, CA, 2006.

[23] B. Zhang, S. Jamin, and L. Zhang. Host multicast: A framework for delivering multicast to end users. In *Proc. of IEEE INFOCOM*, volume 3, pages 1366–1375, 2002.