3DTI Amphitheater: Towards 3DTI Broadcasting

SHANNON CHEN, ZHENHUAN GAO, KLARA NAHRSTEDT, and INDRANIL GUPTA,
University of Illinois at Urbana-Champaign

3DTI Amphitheater is a live broadcasting system for dissemination of 3DTI (3D Tele-immersive) content. The virtual environment constructed by the system mimics an amphitheater in the real world, where performers interact with each other in the central circular stage, and the audience is placed in virtual seats that surround the stage. Users of the Amphitheater can be geographically dispersed and the streams created by the performer sites are disseminated in a P2P network among the participants. To deal with the high bandwidth demand and strict latency bound of the service, we identify the hierarchical priority of streams in construction of the content dissemination forest. Result shows that the Amphitheater outperforms prior 3DTI systems by boosting the application QoS by a factor of 2.8 while sustaining the same hundred-scale audience group.

Categories and Subject Descriptors: C.2.1 [**Computer Communication Networks**]: Network Architecture and Design; C.2.4 [**Computer Communication Networks**]: Distributed Systems

General Terms: Design, Algorithms

Additional Key Words and Phrases: 3D Tele-immersion, broadcasting, content dissemination

ACM Reference Format:
Shannon Chen, Zhenhuan Gao, Klara Nahrstedt, and Indranil Gupta. 2015. 3DTI amphitheater: Towards 3DTI broadcasting. ACM Trans. Multimedia Comput. Commun. Appl. 11, 2s, Article 47 (February 2015), 22 pages.
DOI: http://dx.doi.org/10.1145/2700297

1. INTRODUCTION

Although being more elaborately advanced than conventional 2D video services, existing applications of 3D Tele-immersion (3DTI) are still restricted with a small number of immersive users. While we witness the thriving of live broadcasting video services such as PPLive [PPLive 2014], UStream [Ustream 2014], and Youtube [Youtube 2014], passive nonimmersive audience is rarely included in the design of 3DTI systems. Hence, in Chen et al. [2014], we propose the idea of the 3DTI Amphitheater, a media-enriched multiview live broadcasting system that renders a shared virtual space. 3DTI Amphitheater mimics an amphitheater in the real world (Figure 1) that accommodates a hundred-scale user group. In this article, we provide substantial architectural design of the system and algorithms for adaptive resource allocation regarding user churn (i.e., audience join and leave) in our P2P content sharing scheme.

Users in the 3DTI Amphitheater can be divided into two groups: performers and the audience. A user of an immersive site, or a performer, produces 3D streams by a 3D camera array surrounding her in her own physical user space. A 3D model of the performer will then be constructed from the streams and placed on the virtual stage to

This material is based upon work supported by NSF Grant CNS10-12194, CNS09-64081KN.
Authors' address: University of Illinois at Urbana-Champaign, Siebel Hall, 201 North Goodwin Avenue, Urbana, IL 61801-2302; email: Kcircnc@gmail.com; {zgaoll, klara, indy}@illinois.edu.

ACM1551-6857/2015/02-ART47 $15.00
DOI: http://dx.doi.org/10.1145/2700297

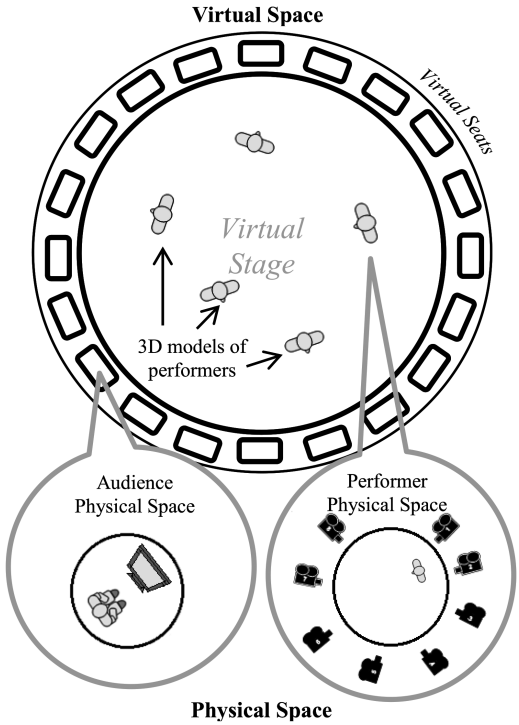


Fig. 1. Concept of 3DTI amphitheater [Chen et al. 2014].

interact with other performers. Thus, the performer crew can be physically dispersed. On the other hand, users of nonimmersive sites, or the audience, passively view the 3D streams created by the performers. They observe the interaction without any involvement. Every performer and every audience is a viewer in the amphitheater since they all need to view the virtual space from their different perspectives. In the virtual amphitheater, a performer's standing position on the virtual stage is synchronized with her position in her physical space so that she can move on the virtual stage freely. The virtual stage and the performer's physical space share the same size. The audience's position is fixed in pre-assigned virtual seats which surround the theater.

3DTI brings new challenges to the well-established IP-based live broadcasting framework. While IPTV service users only perceive one video stream created by one content provider (e.g., ESPN) at any time instance and rely on the service provider (director and cameramen) to choose the view, the 3DTI amphitheater has to support a much more elaborated immersive experience for its users. The first challenge is the multiview feature. The multisource (multiperformer), multicontent (multicamera) dissemination transforms 3DTI broadcasting into a forest construction problem in the P2P network formed by all the participating sites (performers and audiences). Multiple participants subscribe to multiple streams from multiple performers. This introduces massive bandwidth consumption for the dissemination. The 3DTI amphitheater tackles this problem by the design of virtual seats. The audience is placed in virtual seats that surround the central virtual stage and spread evenly, which makes the aggregated view of the audience covers the 360°perspective of a performer. This implies two advantages in the content dissemination: (1) The stream subscriptions of adjacent audiences have substantial overlap due to their similarity of views. Via content sharing in the P2P overlay, these highly overlapped subscriptions can be fulfilled together, and (2) For

each stream capturing a particular facet of a performer, there exists at least one person of the audience who subscribes to it and thus can aid its distribution as a hub. These advantages help us to outperform the prior broadcasting framework [Arefin et al. 2012] with a lower stream subscription rejection ratio under a series of hundred-scale user group simulations. The second challenge is efficient delivery of the multiview content. Although the camera array captures a performer from an omnidirectional perspective, a viewer simply does not require all the streams since she can only fix her view on one side of the performer at any given time. For instance, when the viewer is looking at the front of a performer, the streams capturing the back do not contribute to her view. This leads to the differentiation of streams: to a particular viewer, not all cameras are equally important. For the ease of discussion, we name this the *view-based priority* in the rest of this article.

In addition, we further introduce the differentiation of sites to the dissemination problem. We argue that: to a particular viewer, not all performers are equally important. The importance of a particular performer depends on her role in the performance. The audience may be more interested in the vocalist of a rock band, the diva of an opera, or the quarterback of a football team. Note that the importance of a performer may not be universal. For example, the parents in the audience of a school play must be more interested in their own child on the stage. Thus, we define the role-based priority of a stream to capture its importance based on the semantic relation between its viewer and its performer.

Merging view-based and role-based priorities, we propose the hierarchical stream prioritization. The construction of the content dissemination forest takes the hierarchical priorities as inputs, so that the important streams would not be discarded when the resource is critical.

We evaluate the performance of 3DTI Amphitheater by simulation with real-world network topology compiled by Netmap [Netmap 2014] and the configurations of real 3DTI testbed (TEEVE [Teeve 2009]). We examine the dissemination forest of 3DTI Amphitheater by comparing it with the forest constructed by 4D TeleCast [Arefin et al. 2012], a prior framework of 3DTI broadcasting service. Results show that our framework can sustain the same number of users (100 to 1,000) with a decreased subscription rejection ratio (1,010 more subscriptions admitted while sustaining 1,000 users). The effectiveness of our hierarchical stream prioritization in further evaluated by examining the application quality of service (AQoS) among immersive sites. The AQoS is defined as a weighted admission ratio of stream subscription request, where the weight of a request is determined by the hierarchical priority. Results show that the identification of view-based and role-based priorities helps boost up the AQoS by a factor of 2.8.

The remainder of this article is organized as follows. In the first half of this article, we introduce our system after we review some related works in Section 2. Section 3 contains details of the system components, which classifies the users into three categories. Section 4 introduces the user model, which includes the amphitheater and the hierarchical prioritization. Section 5 introduces the stream delivery model, which contains the pub-sub model and the admission process of stream subscription. In the second half, we first describe the construction and adaptation algorithms of dissemination forest in Section 6, and then we evaluate our system in Section 7. Finally, in Sections 8 and 9, we conclude the article after a discussion of some future directions.

2. RELATED WORKS

3DTI Broadcasting. There are many IPTV frameworks such as Zhang et al. [2005] and Yin et al. [2009] that aim for large scale dissemination of conventional 2D video streams. However, none of them consider multiview dynamics with multisource composition. Therefore, the challenges are different from ours. The 4D TeleCast

framework which targets a large-scale content multicasting of 3DTI is proposed in Arefin et al. [2012]. In order to accommodate the hundred-scale audience group, Arefin et al. propose to allocate the viewer sites into different classes (layers) with different delay service qualities. The higher the service class, the fewer hops there are between the viewer site and the source in the P2P network. The heavy burden of content dissemination not only comes from bandwidth and delay requirements, but also because they tried to support the randomness of the free-viewpoint characteristic. Viewers in the hundred-scale audience can all have their distinct views. This makes the effectiveness of content sharing very unstable in the dissemination network. Thus, the 4D TeleCast has to sacrifice part of the audience by giving them delayed content and requires an additional content distribution network to support the service.

Performer Differentiation. When an interactive multimedia service consists of multiple participants, the importance of content originated from different participants can be different to an observer. In DeVincenzi et al. [2011], the concept is applied on an intelligent teleconference room. The room accommodates multiple participants, and is equipped with a fixed camera that captures the whole room. When a person starts to speak, she triggers the audio sensor near her seat. This notifies the video encoder to allocate higher resolution to the speakers position in the scene.

3DTI Multicast Routing. The differentiation of 3DTI streams introduces extra complexity to the construction of the content sharing overlay. Because every link in the overlay network is transmitting different parts of the stream bundle [Agarwal et al. 2010] (a combination of streams containing different sensing data that are highly correlated), the topology of the network becomes a crucial factor that decides the efficiency of content dissemination. In Yang et al. [2006], a mesh topology is used to deliver the content from each producer site to one another. The number of participating site therefore becomes very restricted (fewer than four). Later, Wu et al. [2008] proposed to alleviate the workload via a randomized admission under a pub-sub model. Yet, the algorithm failed to consider the role of the users and focused only on differentiation at stream level.

3. SYSTEM MODEL

In our system model, multiple 3DTI sites collaborate together to produce and distribute the 3D visual content. Among all the participating sites, only a subset of the sites is producing the 3DTI video streams. Users within these sites have their 3D models projected into the virtual space where they interact with each other. In the rest of the sites, the users only passively observe the activity from the view they choose and do not have their 3D models built by the system. Together, these sites form a P2P overlay network that delivers the content streams. An example of a use case is illustrated in Figure 2(a). Inside the virtual space depicted in the figure, only three performers have their 3D models created (P1, P2, and P3). These performers actively interact with each other through their model on the virtual stage. Around the stage, there are two audiences (A1 and A2) passively observing the interaction from the viewpoint they get assigned. Their sites do not produce any 3D visual contents but only passively receive content from other sites. Based on the different roles of a participating site in a 3DTI session, we classify them into three types: performer sites, audience sites, and the session manager site. In the following, we introduce the system requirements and the assignments of each type.

3.1. Performer Sites

A performer site is a producer of the 3DTI content during a session and also a consumer of the content for other performers. Thus, the hardware requirements of a performer

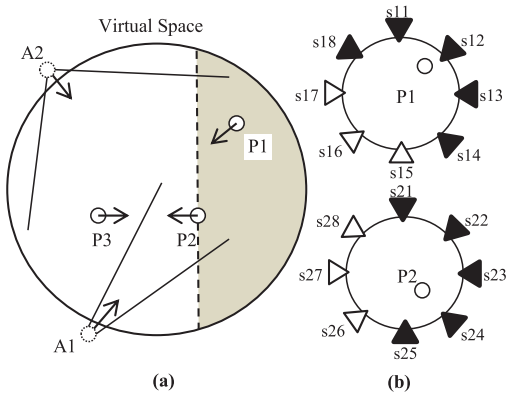


Fig. 2. An example of (a) virtual space and (b) physical space.

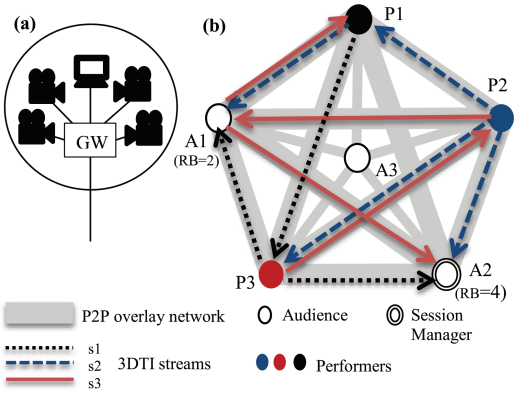


Fig. 3. Content dissemination forest.

site include a 3D camera array, a head mounted display, and a gateway (GW) machine (Figure 3(a)).

The 3D camera array consists of a group of 3D cameras that surround the user space (Figure 2(b)). Each camera captures a facet of the object in the physical user space of the site in form of 3D point cloud. The captured images are stored in a local disk. In real-time, the streams produced by the cameras are delivered to the gateway machine located in each participating site to render the 3D model of the performer. The gateway machines in the performer sites receive streams produced by all the performer sites and render out a consistent virtual space. According to the standing position and the view direction of performer in the physical space, the head mounted display connected to the gateway shows the relative view of the virtual space. For more details, please refer to Chen et al. [2013].

3.2. Audience Sites

The audience sites are observers during a 3DTI session. Users in audience sites passively view the performance without any involvement. Thus, the basic hardware requirement for the audience site only includes the gateway machine and a display. The gateway machines in audience sites collect the streams produced by the performer sites and render the 3D virtual stage space.

3.3. Session Manager Site

In each 3DTI session, one site is chosen by ring-based election [Coulouris et al. 2011] to become the session manager site. The job of the session manager is to examine the subscription requests (Section 5) sent from all the other participating sites. The subscription request contains information to determine (1) which streams the subscriber requires and (2) what the hierarchical priorities (Section 4.2) of the subscriber are. With these requests, the session manager can construct the dissemination network using our forest construction algorithm (Section 6). In reply, the session manager will tell the subscriber from whom should it receive the required streams. More details on this topic will be introduced in Sections 5 and 6.

4. USER MODEL

In this section, we describe our user model in two parts. First, we introduce the characteristics of the Amphitheater and how its structure effects the subscription and dissemination of streams. Second, we introduce the hierarchical stream prioritization.

4.1. Virtual Amphitheater

The virtual space mimics an amphitheater, where the performers are interacting on the central virtual stage and the audiences are assigned with their own virtual seats which surround the stage and disperse evenly. Thus, the perspectives of a performer and an audience are different. While a user in a performer site may not be able to see the whole performer crew due to her standing position and view direction, the users in the audience site can always choose to capture the whole view of the stage (zoom-out) or to focus on part of the stage (zoom-in). As illustrated in Figure 2(a), where the arrows indicate the view directions, performer P2 can only see P3 but not P1 since P1 is standing in her blind side (the grey area). As for A1 and A2 in the audience, A2 choose to see the whole stage while A1 focuses on a particular part as if she sees through a pair of opera glasses.

The virtual seats fix the position of each audience. Although this design restricts the audiences from moving their viewpoints freely inside the whole virtual space (e.g., around stage or even on stage), it complies with the common sense in a real theater, where seats are pre-assigned and fixed during the performance. Furthermore, the design brings two advantages to the delivery of streams. First, it enhances the effectiveness of content sharing. For two audiences in adjacent seats, their views are very likely to overlap with each other by a fair portion. This implies that in stream delivery, the same stream is more likely to be subscribed by multiple sites, which makes the sharing of the content being able to save more bandwidth in the P2P network.

Second, the surrounded arrangement of seats helps the distribution of streams from their performer sites. Since the stage is surrounded by the audience, each and every side of the performers body must be looked at by some audiences. This implies that every stream (each capturing different sides of the performer) is likely to be subscribed by one or more audience sites at any time. In the dissemination network, these audience sites act like hubs that help the performer sites to distribute their streams to other performers. Often times, the outbound bandwidth of a performer site is not enough to distribute its streams to all the other performers when the performer crew grows bigger In that case, audience sites can help as hubs to relay those contents. The surrounded seat design raises the possibility of the existence of such hub audiences.

4.2. Hierarchical Stream Prioritization

The view-based priority and the role-based priority are both addressed in the hierarchical prioritization of streams. We introduce the three logical objects that we define in the hierarchy as follow (using Figure 2 as an example).

—*Stream.* A 3D video stream created by a camera. This is the basic content unit in the dissemination network. We denote a stream by s with a postfix number for identification, for example, $s22$.
—*View.* The set of streams that are created in the same performer site. We denote a view by v with a postfix number to identify the site, for example, $v2 = \{s21, s22, s23, s24, s25, s26, s27, s28\}$ is the stream set generated by performer site P2 in Figure 2.
—*Session.* The set of views in the amphitheater. We denote a session by x, for example, $x = \{v1, v2, v3\}$.

As we discussed previously, there are two factors that affect the importance of a stream to a particular viewer. First, the view-based priority reflects the importance of a camera per site. Second, the role-based priority reflects the importance of performer per session. If a stream captures a performer in whom the viewer is more interested, then the stream is more important to the viewer.

To address these factors, each viewer would provide information to determine her own hierarchical priority in her subscription request (Section 5). The hierarchical priority is represented as a sequence of numbers assigned to each element in a view or a session. For example, to address the view-based priority, a viewer may set her hierarchical priority as

$$HP(v2) = \{0, 0, 0, 0, 3, 4, 3, 0\} \tag{1}$$

$$v2 = \{s21, s22, s23, s24, s25, s26, s27, s28\}. \tag{2}$$

This hierarchical priority (HP) states that, for this viewer, stream $s26$ is the most important stream among all streams in the set $v2$; $s25$ and $s27$ come second; and the viewer does not care about $s21$ to $s24$ and $s28$. A larger number indicates higher importance. To determine the numbers for view-based priority, we modify the contribution factor (CF) proposed by Yang et al. [2006]:

$$CF = \vec{O}_i \cdot \vec{O}_u, \tag{3}$$

where $\vec{O}_i$ is the shooting direction of a camera, $\vec{O}_u$ is the view direction, and CF is defined as their inner dot product. Our priority numbers are calibrated from the CF value by (1) treating nonpositive CF as zero and (2) normalizing the numbers so that their sum becomes ten. The calibration is only for the ease of computation in the construction of the dissemination forest (Section 6).

To address the role-based priority, the hierarchical priority is defined similarly, as a mapping from the views in a session to numbers, for example,

$$HP(x) = \{5, 3, 2\} \tag{4}$$

$$x = \{v1, v2, v3\}. \tag{5}$$

This states that, for the viewer who assigned this HP, the view-capturing performer #1 ($v1$) is more important than $v2$ and $v3$; and $v3$ captures the least important performer. Again, the numbers are nonnegative and they are normalized so that their sum is ten for the ease of computation. The determination of the numbers depends on the role of the viewer and the scenario. Here we provide three examples.

Viewer Role 1. Parent in the audience of a school play.

$$HP(x) = \{1, 8, 1\} \tag{6}$$

$$x = \{v1, v2, v3\}. \tag{7}$$

The priority numbers are subjectively assigned by the viewer (parent). Since $v2$ is capturing the image of the viewer's child on stage, she assigns the highest priority to it.

Viewer Role 2. Player in a table tennis dual match.

$$HP(x) = \{3.33, 3.33, 3.33\} \tag{8}$$

$$x = \{v1, v2, v3\}. \tag{9}$$

In this case, the viewer is a performer. The priority numbers are uniform. For $v1$ being the viewer's teammate and $v2$, $v3$ being the opponents, they have equal possibilities to change the game. Thus, the role-based priority numbers are the same for all three views.

Viewer Role 3. Participant of a cocktail party.

$$\begin{aligned} HP(x) = \{&1/\text{distance to performer\#1}, \\ &1/\text{distance to performer\#2}, \\ &1/\text{distance to performer\#3}\} \end{aligned} \tag{10}$$

$$x = \{v1, v2, v3\}. \tag{11}$$

In this scenario, all the participants (performers) interact with each other in a ball room (stage) that is monitored by the security (audience). The priority numbers are objectively determined based on the distance between two participants in the virtual environment. This way, the view of a performer who stands closer to the viewer will be given higher priority than the ones that are far away. Higher priority implies higher admission rate of streams in the dissemination forest construction (Section 6) and thus a higher quality of the performers image. This complies with the sense in the real world, where it is hard to see clear of a further object but easy to see an object clearly when it is closer.

The three examples show three possible ways to determine the role-based priority: (1) user-defined, (2) uniform, and (3) objective. Note that these are not the only methods but rather intuitive examples. Depending on the characteristic of scenarios and roles, we expect more creative determinations so that our system can become TEEVE (Tele-immersive Environment for EVErybody).

5. STREAM DELIVERY MODEL

Our Amphitheater system adopts the pub-sub (publish-subscribe) model [Eugster et al. 2003] as its content dissemination paradigm. The model has three core components: publisher, subscriber, and broker. In the Amphitheater, the publishers are the performer sites; the subscribers are all the viewers (including both performer and audience sites); and the broker is the session manager site. We introduce the message exchange among the three components as follows. An illustration of the whole process is shown in Figure 4.

In the beginning of a session, the publishers register their cameras to the broker. The registration states the number of cameras that the performer owns and their shooting angles. The subscribers, on the other hand, submit their subscription requests to the broker, which contains the type (performer or audience), the state (position and view direction), and the interests (user-defined role-based priority) of itself. Whenever there is an update on the subscribers information (e.g., change of position), it renews the subscription request and sends it to the session manager again. Registrations and subscriptions also contain the geographic location and the maximum inbound/outbound bandwidth of the sender. This information is used to estimate the propagation delay between sites and will be used in the construction of dissemination forest (Section 6).

After the broker receives all the registrations from the publishers and all the subscription requests from the subscribers, it starts to translate them into stream requests. Since the broker knows (1) the positions and view directions of every viewer, and

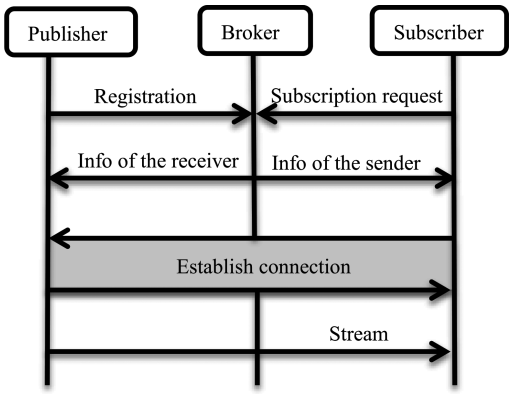


Fig. 4. Protocol in the pub-sub model.

(2) the positions of each performer and the shooting angles of their streams, it can deduct which streams would be needed by a particular subscriber in order to construct her view. For example, in Figure 2, after the broker gets the positions of all participants, it will know that stream $s26$ of performer P2 is needed by audience A1.

Furthermore, the broker would know how important a stream is to the particular viewer by considering its view-based and role-based priorities. Since the broker knows the position and view information of each viewer-performer pair, it can infer the view-based priority for each stream request. In Figure 2, for audience A1 who has a 45°view direction (we set 0°to be pointing to the right), the CF of $s26$ is $\cos 0 = 1$ because $s26$ also shoots with a 45°direction. CF of $s25$ for A2 is $\cos(\pi/4) = 0.707$ because there is a 45°difference between A2's view direction and $s25$'s shooting direction.

The determination of role-based priority, on the other hand, depends on how it is defined as we discussed earlier in Section 4.2. For the first two determination methods: user-defined and uniform role-based priorities, the broker needs no extra computation to get the priority numbers since they are already specified in the subscription request (the interest field). As for the objective determination method which is based on distance, the computation requires the positions of the viewer and the performers in the virtual space. This information is already stated in the subscription requests (the state field). Thus, the role-based priorities of all the subscribers are also known by the broker.

After the stream requests and the hierarchical priorities are determined, the final job of the broker is to construct the content dissemination forest and then to notify the subscribers from whom should they receive the streams. We leave the construction details for the next section. Note that a subscriber does not necessarily receive streams from a publisher as depicted in Figure 4. Instead, it may receive the stream from another subscriber who needs the same content because of P2P sharing.

6. DISSEMINATION FOREST

The content dissemination forest decides the efficiency of stream delivery. Under the pressure of massive audience population, bandwidth-consuming streams, and tight delay bound, an efficient dissemination network would help preserving the service quality with the limited resource.

Now we continue the job of the broker (session manager) after it acquires the stream requests and the hierarchical priorities of the viewers. The objective is to construct a set of directed trees (a forest) in the P2P network among the participating sites.

Each tree connects all the subscribers who require the same stream with the publisher (performer) of that stream as the root. An example is provided in Figure 3(b). We simplify the situation by assuming each performer only produces one stream. Under resource limitations, often times not all the stream requests can be admitted. Thus, construction of an efficient forest which preserves a low request rejection ratio is crucial.

Another important metric that reflects the efficiency of a forest is its resulting application quality of service (AQoS). AQoS is a weighted version of the admission ratio of stream requests (formal definition will be provided in problem formulation). From hierarchical priorities, we know the importance of a particular stream to its subscriber. The more important the stream is, the higher its weight should be. Thus, the AQoS metric essentially shows whether the forest construction algorithm can identify the importance of the content and can choose the less important ones to discard when there is not enough resource. In the following, we first formulate the forest construction problem and then introduce our construction dissemination algorithms. The first forest construction algorithm handles the initial construction. Since our use case mimics a theater play in reality, we can assume that most of the audience join the audience together from the beginning of the performance. Thus, the algorithm collects priorities from these initial audiences and constructs the initial dissemination forest. The second forest adaptation algorithm adjusts the structure of the forest to handle user churn. Since audience may leave and join during the performance, the algorithm maintains the efficiency of the P2P sharing under the dynamics.

6.1. Problem Formulation

The problem contains two constraints and two optimization goals.

Bandwidth Constraint. For each participating 3DTI site U, there is a limit on the inbound and outbound bandwidth of its local gateway machine, denoted as I_U and O_U. These limits can be measured by various probing tools (e.g., Pathload [Jain and Dovrolis 2003]). The total bandwidth consumed by streams received by the site must not exceed the limit I_U; and the total bandwidth consumed by streams going out from the site (including those produced by the site itself and those relayed by the site) must not exceed the limit O_U.

Latency Constraint. To preserve the real-time property of 3DTI service among performer sites, an end-to-end latency bound DI is placed on the content delivery to a performer site. As for audience sites, since a delay of content delivery in a reasonable range ($<$5s) does not degrade the service quality, the latency bound for audience sites, denoted as DP, is typically larger then DI. The P2P overlay network connecting participating sites is a complete graph with a cost on each of its edges. The cost denotes the time delay for a stream to travel from one end to the other. These costs are estimated by the geographic locations of the sites by the empirical mapping provided in Feldman and Shavitt [2007]. In forest construction, the total cost of a delivery route must not exceed the latency bound.

Minimizing the Stream Request Rejection Ratio. Any stream request that violates either one of the constraints will be rejected by the session manager site at forest construction. Since every request rejection implies potential degradation of the service quality, our first goal of the construction is to minimize the number of rejected stream requests. Thus, we define the *request rejection ratio* of streams as the number of rejected stream requests (Nr) over the total number of stream requests (N): Nr/N.

Maximizing the Application Quality of Service (AQoS). With the view-based and role-based priorities of a subscriber defined in Section 4.2, we define the hierarchical priority (hp) of a stream request as the product of (1) the priority number of the stream in view-based priority, and (2) the priority number of the view that contains the stream

in role-based priority. For example, if for a particular viewer

$$HP(v2) = \{0, 0, 0, 0, 3, 4, 3, 0\} \tag{12}$$

$$v2 = \{s21, s22, s23, s24, s25, s26, s27, s28\}, \tag{13}$$

and

$$HP(x) = \{1, 8, 1\} \tag{14}$$

$$x = \{v1, v2, v3\}, \tag{15}$$

then the *hp* of stream $s25$ is $3 \times 8 = 24$ (HP($v2$) of $s25 \times$ HP(x) of $v2 = hp$). Since the priority numbers (elements of HP()) range from zero to ten, the resulting *hp* ranges from 0 to 100. Note that if either of the priority numbers is zero, it means (1) this stream does not contribute to the view at all, or (2) the subscriber has no interest in the performer captured by this stream. In either case, the resulting $hp = 0$ signifies the session manager to ignore this stream request. We define the AQoS as

$$AQoS = \frac{\text{sum of } hp \text{ of the admitted requests in the session}}{\text{sum of } hp \text{ of all requests in the session}}. \tag{16}$$

6.2. Initial Forest Construction

Wang and Crowcroft [1996] proved that when a multicast routing problem is bound with two or more orthogonal constrains (in our case: bandwidth and latency), it becomes a NP-complete problem. Thus, we propose a heuristic solution based on the *hp* of the stream requests. The main complication of our forest construction problem is two-fold. (1) Which request should be examined by the session manager first? and (2) From whom should the subscriber receive the stream when there are multiple holders of the requested content?

Order of Request Examination. The order of stream request examination decides the possibility for a specific request to be admitted. Intuitively, the first request being examined should always be admitted since the bandwidth of the overlay network has not been occupied by any other delivery. The later a request is examined indicates the higher the chance should it be rejected since the links could be occupied by preceding requests. Thus, we order the stream requests of all the subscribers by the *hp* of the requested stream from high to low.

Selection of Sender Site. Since a stream is shared among the subscribers in the same tree, there can be more than one site caching the same stream. For example, in Figure 3(b), if A3 has a request of stream s1, the potential sender sites are P1, P3, A1, A2 since they all hold s1. Our selection strategy can be broken down to three heuristics: similarity, residual bandwidth, and distance, ordered by the sequence of application.

Heuristic 1: Similarity. According to the type (performer or audience) of the subscriber, it will be assigned to the sender of the same type if possible. This helps the performer sites receive streams from another performer. The end-to-end latency among performers can be reduced because the performer sites will be closer to the root (the source performer) in dissemination trees. Since the audience sites would also receive streams from their own kind over the performer sites under this heuristic, the outbound bandwidth of performer sites can be reserved for other performers.

Heuristic 2: Residual Bandwidth. A subscriber will be assigned a sender with the most residual bandwidth. For an audience site, the residual bandwidth equals to its maximum outbound bandwidth O_U subtracted by the bandwidth consumed to share streams. As for performer sites, the session manager has to make sure that all the requested streams can be sent out from its performer to at least one other site, or else no subscriber can receive this stream. Thus, the residual bandwidth is further

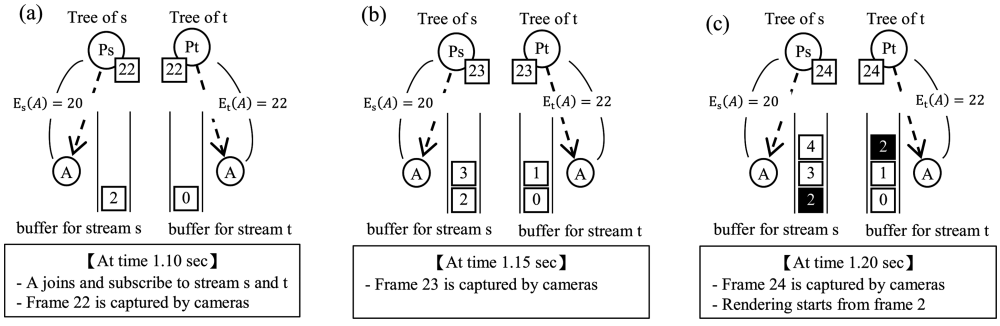


Fig. 5. Stream synchronization.

deducted by a reserved bandwidth. The reserved bandwidth of a performer site is computed as the sum of bitrate of streams that fits the following three conditions: (1) The stream is produced by the performer itself; (2) The stream is requested by at least one subscriber; (3) The stream has not been sent to any other site yet. This strategy is originally proposed in Wu et al. [2008], which guarantees that a requested stream can be disseminated before the outbound bandwidth of its producer is saturated.

Heuristic 3: Distance. The sender is set to be the one who is closer to the root in the dissemination tree. This shortens the end-to-end delay.

We now use Figure 3(b) as an example to demonstrate the selection of sender site. We simplify the problem in this example by assuming the inbound and outbound bandwidths of all sites are able to sustain no more than four streams (assuming homogeneity of stream bitrates). The session involves six sites in total, which includes three performer sites (P1, P2, P3) and three audience sites (A1, A2, A3). The RB in the figure indicates residual bandwidth (in number of streams it can sustain). Following the order of request examination, let the next three stream requests to be examined by the session manager to be "A3 requesting $s1$", "A3 requesting $s2$", and then "A3 requesting $s3$". Thus, by our selection strategy, A2 will be the assigned sender for $s1$ because it is the audience site (by Heuristic 1) with the most residual bandwidth (by Heuristic 2). After this assignment, the RB of A3 becomes three. Next, A2 will also be the sender for $s2$ for the same reason and RB of A3 becomes two. Finally, the sender of $s3$ will be A1 because now the two audience sites (by Heuristic 1) have the same residual bandwidth (by Heuristic 2) but A1 is closer to the root of the tree that disseminates $s3$ (by Heuristic 3).

6.3. Forest Adaptation

After the initial forest is constructed following the previous algorithm, audiences can still leave and join the Amphitheater. Thus, in order to preserve the efficiency of resource utilization, a forest adaptation algorithm is proposed to handle this audience churn. Before we dive into details of the algorithm, a few notations and concepts must be introduced for the ease of explanation. First, we define notation $A1 \xrightarrow{s} A2$ to denote that audience 2 is downloading stream s from audience 1. Note that, this relationship implies A2 will never get a particular frame of stream s before A1 gets it because each audience only downloads a stream from one source.

Second, since stream delivery introduces delay, an audience may not be able to get the newest frame captured by the camera instantaneously. In other words, there exists a "frame elapse" between the newly captured frame at the performer site and the frame being played out at the audience site. For example, in Figure 5(a), if it takes 1 second for stream s to be delivered to audience A and the frame rate is 20fps, then at the time

the ith frame is captured by performer Ps, A is still playing frame $i-20$. Thus the frame elapse is 20 and we denote it as $E_s(\mathrm{A}) = 20$. Note that, since all the subscribed streams have to be synchronized at the audience site so that they can be rendered together, the frame elapses of each stream have to be adjusted to be the same. For example, if A also subscribes to stream t and it takes 1.1 seconds to deliver ($E_t(\mathrm{A}) = 22$, Figure 5(a)), then the audience has to wait for 2 frames of stream s before it can render the two streams together (Figure 5(c)). From Figure 5, we see that when the rendering finally happens, the frame elapse between the rendered content and the frame captured is $22 = 24 - 2$. Thus, we define this adjusted frame elapse for an audience to synchronize all her subscribed streams as $E(\mathrm{A}) = \max_{s \in S'}\{E_s(\mathrm{A})\}$, where S' is the set of streams subscribed by audience A.

From the previous two notations, we can determine that the following is always true:

$$\mathrm{A1} \xrightarrow{s} \mathrm{A2} \Rightarrow E(\mathrm{A1}) \leq E(\mathrm{A2}). \tag{17}$$

Now we are ready to introduce the planning algorithm. The algorithm is two-fold, each handles audience join and leave.

6.3.1. Audience Join. From the example illustrated in Figure 5, we know that the time spent to wait for the subscribed streams to be synchronized contributes to the delay time for audience join event. In the example, the time spent is 100ms (waiting time for two frames). We call this the synchronization delay. In formal form, the synchronization delay is determined by

$$sync\_delay = \max_{A1, A2 \in A'} |E(\mathrm{A1}) - E(\mathrm{A2})| / frame\_rate, \tag{18}$$

where A' is the set of audience sources who share streams to the newly joined audience A, that is,

$$A' = \{\mathrm{As} | \mathrm{As} \xrightarrow{s} \mathrm{A}, s \in S'\}, \tag{19}$$

and S' is the set of streams to which A subscribes.

Hence, the objective of this algorithm is to minimize the difference between frame elapses of chosen sources. A naive policy would be to let the new audience always download streams directly from the performer. This way, the new audience will always get the latest frames and the frame elapses of sources are likely to be similar. However, this policy will soon saturate the outbound bandwidth of performer sites, which can easily violate the bandwidth constraint. Thus, the new audience should always download a stream from an existing audience if possible. In the following, we first show the pseudocode of the algorithm then provide explanation afterwards.

The algorithm is comprised of two phases. In phase one, we first select the existing audience who holds the newest content in each dissemination tree of each stream that the new audience intends to subscribe (Line 2 to 12). For example, in Figure 6(a), $S' = \{s_a, s_b, s_c\}$ and the dissemination trees are illustrated. The roots are performers and the other nodes are existing audiences. In this example, the audiences who hold the newest content in each tree would be A1, A2, and A3, respectively. Note that the selected audiences must have available bandwidth to send stream to the new audience. If not, we select the one with the second newest content and so on. If there is no audience in a tree who has available bandwidth to share, then the new audience will directly download the stream from the performer (Line 6 to 8). This part of the algorithm is to ensure the liveliness of content. We want the new audience to be downloading the newest content possible.

Next, we pick the audience who holds the oldest content among the selected audiences and set up an interval I that contains the frame elapse of this audience (Line 16 to

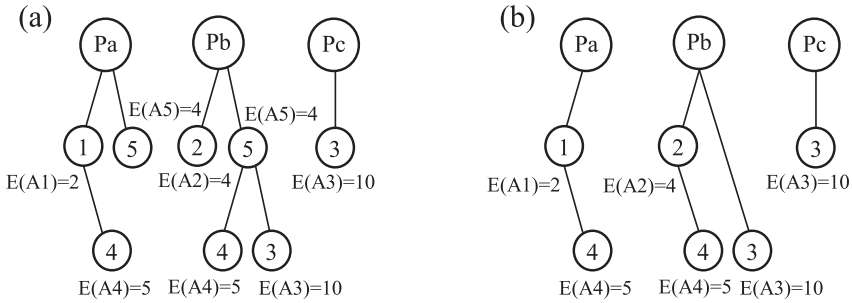


Fig. 6. Forest adaptation.

ALGORITHM 1: UJoin(A, S')

$C \leftarrow \emptyset$
foreach *s in* S' **do**
  Find (As, s) which has the smallest E(As) such that
  (1) audience As subscribes to stream s
  (2) audience As has available bandwidth
  if *no such* (*As*, s) *exists* **then**
    Let new audience A download stream s from the performer
    Remove s from S'
  else
    Add (As, s) to C
  end
end
if $C = \emptyset$ **then**
  return
end
($\check{A}s$, $\check{s}$) $\leftarrow$ audience $\check{A}s$ who has the largest E(As) in C
Let new audience A download s from $\check{A}s$
Remove $\check{s}$ from C
$I \leftarrow [E(\check{A}s), E(\check{A}s)]$
$C \leftarrow \emptyset$
while $S' \neq \emptyset$ **do**
  foreach *s in* S' **do**
    Find (As, s) which has E(As) closest to I such that
    (1) audience As subscribes to stream s
    (2) audience As has available bandwidth
    if *no such* (*As*, s) *exists* **then**
      Let new audience A download stream s from the performer
      Remove s from S'
    else
      Add (As, s) to C
    end
  end
  ($\check{A}s$, $\check{s}$) $\leftarrow$ audience $\check{A}s$ who has the smallest E(As) in C
  Let new audience A download s from $\check{A}s$
  Remove $\check{s}$ from C
  Include $E(\check{A}s)$ to I
end

19). In Figure 6(a), this audience would be A3 and interval $I = [10, 10]$. Now we enter phase two. In this phase, the goal is to minimize the difference between frame elapses of sources by keeping interval I small. Interval I covers all the frame elapses of all chosen audience sources, that is,

$$A' = \{\text{As}|\text{As} \xrightarrow{s} \text{A}, s \in S'\} \tag{20}$$

$$I = \left[\min_{A \in A'} E(\text{A}), \max_{A \in A'} E(\text{A})\right]. \tag{21}$$

Hence,

$$sync\_delay = (|I| - 1)/frame\_rate. \tag{22}$$

Therefore, minimizing the size of the interval becomes the objective of the rest of the algorithm (Line 20 to 37). In each dissemination tree, we find the audience who has a frame elapse that is closest to interval I (Line 23). We let the new audience download the respective stream from this audience and we update I by including its frame elapse (Line 36). Continuing the example in Figure 6(a). Now that $I = [10, 10]$ and $S = \{s_a, s_b\}$, in the next iteration A0 will be assigned to download s_b from A3 since $E(\text{A3}) = 10$ is the closest to I, which makes $I = [10, 10]$ and $S = \{s_a\}$. Finally, A0 will be assigned to download s_a from A4 since $E(\text{A4}) = 5$ is the closest at this point, which makes $I = [5, 10]$, $S' = \emptyset$, and the synchronization delay is 4 divided by the frame rate.

6.3.2. Audience Leave. Audience leave events can be classified into normal leave or abnormal leave. Normal leave is when the leaving audience notifies the session manager before it disconnects, so that the manager can handle the children of the leaving audience in the dissemination trees. Abnormal leave is caused by unexpected termination of the audience site. In this case, the children have to detect the incident and notify the session manager to be reassigned a parent. Detection of parent failure is done by standard heartbeat approach [Coulouris et al. 2011] by treating the incoming frames as keep-alive messages.

Thus, for the session manager, there is no difference between normal and abnormal leaves since it will always be notified, either by the leaving audience itself or its children. The adaptation algorithm only need to handle the reassignment of parent whenever an audience loses its source. The following is the pseudocode.

ALGORITHM 2: Reassign(O,S'_O)

```
foreach s in S'_O do
    Find (A, s) such that
    (1) E(A) + ε = E(O)
    (2) audience A subscribes to stream s
    (3) audience A has available bandwidth
    if no such (A, s) exists then
        Let orphan O download stream s from the performer
    else
        Let orphan O download stream s from audience A
    end
end
```

Input of Reassign( ) are an orphan O and a set S'_O which contains the streams O subscribes that no longer have sources to download from. Note that, although the code is similar to phase one of UJoin( ), the conditions on source selection change (Lines 2 to 5). We want the transition for an orphan to its new parent to be seamless without any interruption in its video playout. Thus, we have to find a new parent who is holding

content that, after propagation delay (ϵ/*frame_rate*) between orphan and itself, can continue with the same frame elapse as the orphan (Line 3). The probability of finding such parent is fair since the system is targeting to serve a large audience group. However, if unfortunately such new parent does not exist, the orphan is reassigned to the performer. As we mentioned in Section 4.2, performer sites store all the frames captured during the performance. Thus, a performer site can emulate any frame elapse by providing old content to the orphan.

Transition of orphan reassignment is illustrated from Figure 6(a) to 6(b). Here we use $\epsilon = 1$. If A5 leaves, it makes A3 and A4 orphans. After A3 requests to be reassigned, it will download s_b directly from the performer since no existing audience can continue its frame elapse. After reassigning A4, it will be downloading s_b from A2 (Figure 6(b)).

7. EVALUATION

The evaluation of our system is four-fold. First, we evaluate the overall performance of the Amphitheater with hundred-scale audience group. Second, we focus on the service quality of the performer sites. Third, we verify the effectiveness of virtual seat design on improving the efficiency of stream dissemination. Last, we focus on practical P2P issue by measuring the delay caused by user churn. In the following sections, we first introduce the settings of our experiment testbed before we continue to the evaluations and their result analysis.

7.1. Simulation Settings

Network Settings. We adopt real-world topology from Netmap [Netmap 2014] as the testbed of our simulation. Among the 1,092 actual hosts distributed around the world in the Netmap database, we randomly picked 3 to 1,000 hosts to be our participating 3DTI sites. The host-to-host delay is estimated based on the geographic distance between them [Feldman and Shavitt 2007]. We set the maximum inbound (I_U) and outbound bandwidths (O_U) of a site to be random values in the range of 40 to 150Mbps based on our observation of the TEEVE system.

Site Settings. Each participating performer site is equipped with a camera array with eight 3D cameras shooting from octagonal positions around the user space. According to our observation of the TEEVE system, each camera produces a 3D video stream with a 5 to 10Mbps bitrate. Hence, in the simulation, the bandwidth consumption of a stream is set to be a random number in that range.

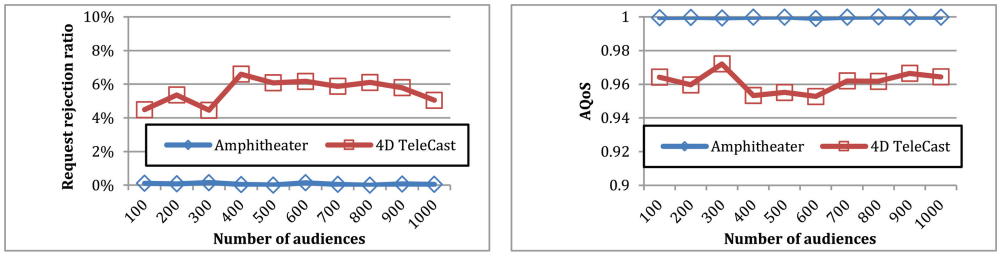
Latency Settings. The latency bound is set to be 100ms (DI) for performer sites and 5s (DP) for audience sites. Contents should be delivered with an end-to-end delay exceeding the bound will not be admitted by the session manager.

7.2. Broadcast with Hundred-Scale Audience

In this part of the evaluation, we evaluate the overall performance of the Amphitheater as a hundred-scale broadcasting system of 3DTI. We investigate the effect of the size of performer crew and the size of the audience in the two parts. The simulation results are compared with 4D TeleCast [Arefin et al. 2012], a prior 3DTI dissemination system that also targets on large scale broadcasting.

Performer Settings. On the circular stage of the Amphitheater, we randomly assign the standing positions and the view directions of the performers. The role-based priorities are set according to the distance between the viewer and the targeted performer (Viewer Role example 3 in Section 4.2).

Audience Settings. The audience is put in virtual seats, which surround the stage and are evenly dispersed. We set the views of the audiences to be pointing towards the center of the stage and the field of view covers the whole stage (the same as audience A2, depicted in Figure 2(a)). The role-based priorities are set according to the popularity



(a) Request rejection ratio **(b)** AQoS

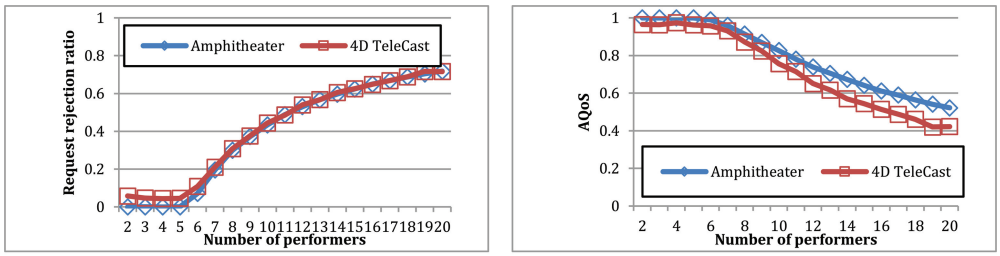
Fig. 7. Broadcast with fixed number of performer sites.

of the targeted performer. In the performer crew, we set 25% of the performers as being popular, 50% of them being average, and 25% of them being subordinate. The role-priorities assigned by the audience are random variables. However, we set the average value of the assigned role-priorities to be the highest for popular performers; and the lowest for subordinate performers. This setting is a more sophisticated version of the Viewer Role example 1 in Section 4.2.

Simulation Results. In the first part, we fix the number of performer sites at five and gradually increase the total number of audience sites from 100 to 1,000. The results are plotted in Figure 7 with the performance of the Amphitheater being the blue-diamond curve and 4D TeleCast being the red-square curve. We can see that the number of the audience sites does not affect the rejection ratio significantly. Intuitively, more sites participating in the session implies taller stream dissemination trees (since the outbound bandwidth is limited) and hence longer latency. However, since noninteractive audience has a much higher tolerance towards delay ($DP = 5$s), the growth of the audience group does not increase the stream request rejection ratio in both of the frameworks. The rejection ratios are lower than 10% for both cases and the AQoS are higher than 0.9 due to the high request admission rate.

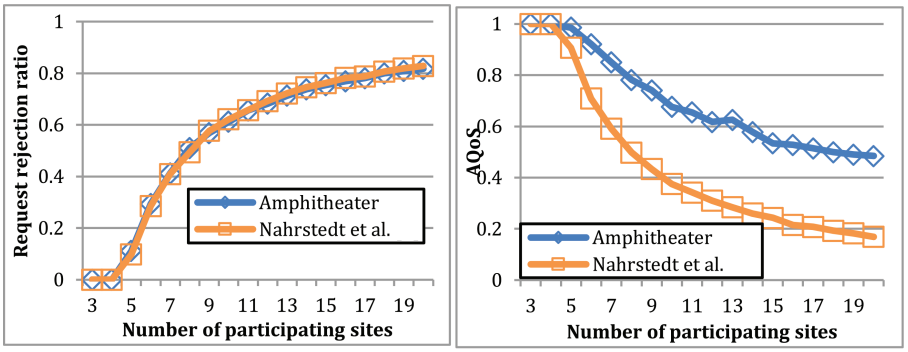
Comparing the performance of the two frameworks from Figure 7, we can see that even with abundant resource, 4D TeleCast still has a slightly inferior performance. The phenomenon is caused by the "number of accepted stream constraint" in the design of 4D TeleCast. Since role-based priority is not identified in 4D TeleCast, its scheduling algorithm assumes equal importance of all performer sites and will drop a viewer (by rejecting all of its stream requests) when it cannot receive at least one stream from each performer. On the other hand, since our examination order of stream requests inherits the role-based priority, if a viewer does not receive any stream from a particular performer, it is because she has marked that performer as unimportant with a low role-based priority number. The all-or-nothing design of 4D TeleCast oversees the fine-grained hierarchical priority of streams. This contributes to an inferior performance when the user group grows. When the size of audience reaches 1,000, the Amphitheater sustains 1,010 more stream requests than 4D TeleCast with higher AQoS.

In the second part, we conduct another set of simulation with a fix number of audience (500 sites) and performer crews with different sizes (two to twenty). The results are shown in Figure 8. We can see from the figure that our dissemination forest is able to identify the important requests and to assign resource accordingly. With no significant difference ($<$6%) between the two frameworks on request rejection ratios, our algorithm is able to reach a higher AQoS. The improvement from adopting our algorithm grows with the increasing pressure of resource limitation. When the number of performers reaches twenty, the total bitrate of streams created by the whole performer crew is 1,600Mbps, and our algorithm achieves 24% more AQoS than 4D TeleCast.



(a) Request rejection ratio **(b)** AQoS

Fig. 8. Broadcasting with fixed number of audience sites.



(a) Request rejection ratio **(b)** AQoS

Fig. 9. Virtual play.

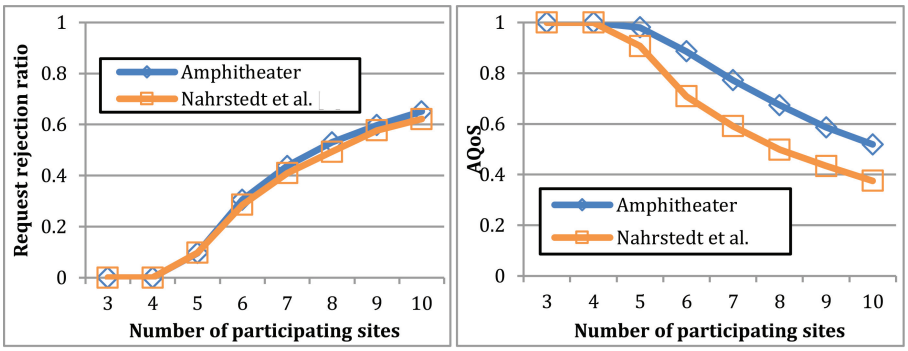
7.3. Service Quality of Performers

In this part of the evaluation, we focus on the performer sites and evaluate their service qualities. The quality of a performance depends largely on the quality of interaction among the performers on stage. We compare our result with the forest construction algorithm used in the framework proposed by Nahrstedt et al. [2011], which assumes every participating site is immersive and hence every user (in a small user group) is a performer in their scenario. Thus, in the simulation, we accommodate the Amphitheater to this scenario by setting the auditorium to be empty. There are only interactive users in the session.

Performer Settings. The evaluation contains two parts. First, we set the stage at a virtual play, which contains fewer than twenty performers, and the role-based priority of a viewer is set according to the distance (Viewer Role example 3 in Section 4.2). Second, we set the stage at a sport arena. Where fewer than ten performers (athletes) are in the arena and the role-based priority of a viewer is set to be uniform (Viewer Role example 2 in Section 4.2). In both scenarios, the performers are placed in random positions on the stage and each of them has a randomly set view direction.

Simulation Result. Since all participating sites are performer sites, intensive stream exchanges and hence massive bandwidth consumption in the overlay network are well expected. The results of virtual play and sport arena are presented in Figure 9 and 10, respectively.

In Figure 9(a) and 10(a), the request rejection ratios of the two algorithms are plotted against the number of participating sites with the results of the Amphitheater being the blue-diamond curve and Nahrstedt et al. [2011] being the orange-square curve.



(a) Request rejection ratio **(b)** AQoS

Fig. 10. Virtual sport arena.

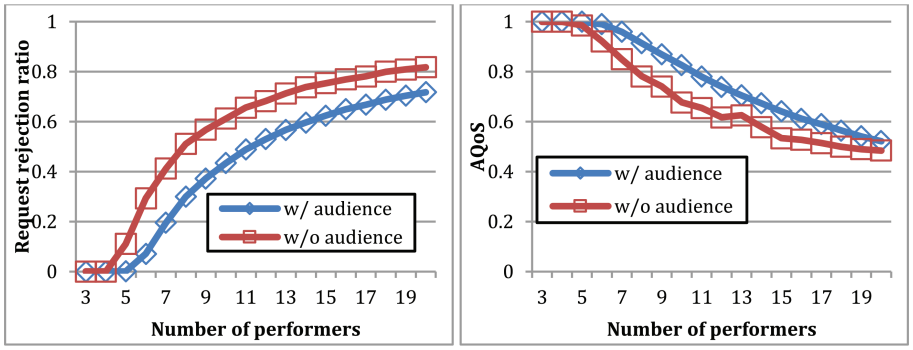
First, we can see the ratios increase along with the number of the participating sites in both cases. With the constant networking resource, the increasing number of sites is introducing more stream requests that cannot be admitted due to the lack of bandwidth or to the violation of latency constraint. From the figure, we can see that for cases with fewer than five participating sites, the network can sustain all of the requests addressed by the users. After that, the rejection ratio rises gradually. The rejection ratio is nearly 80% when the number of sites reaches twenty in the virtual play; and 60% when it reaches ten in the sport arena.

Second, comparing the performance of the two systems, we can see the resulting rejection ratios are very similar. Nahrstedt et al. [2011] adopt the random join algorithm to construct their content dissemination forest. The basic idea is to examine the stream requests in random order to avoid biased resource allocation to any content producer or subscriber. In our algorithm, the hierarchical priority provides enough randomness to the order of stream request examination. The standing position of a performer in the user space (which affects the view-based priority) and the different roles of the performers (which affect the role-based priority) are independent of the underlying resource demand. Hence, the *hp* of a stream request, defined as the product of view-based priority number and role-based priority number, provides sufficient randomness for the low-level resource allocation. In other words, the *hp* value is no different from a random value in forest construction. Hence, the two algorithms achieve similar request rejection ratios in all cases (difference $<4\%$).

Last, we look at the AQoS of the two algorithms in Figures 9(b) and 10(b). Since AQoS is actually a weighted version of the admission ratio of stream requests, the curves of AQoS have opposite shapes of rejection ratio. As the number of sites increases, AQoS drops intuitively. Comparing the two algorithms, we can see that our algorithm outperforms [Nahrstedt et al. 2011] by a factor of 2.8 when the number of sites reaches twenty in the virtual play, and 1.4 when the number of sites reaches ten in the sport arena. This shows that, although the two systems has rejected the same amount of stream requests (Figures 9(a) and 10(a)), our Amphitheater can identify the important streams and assign higher priorities to them when the resource becomes scarce.

7.4. Effect of Virtual Seats

In this section, we verify the effectiveness of virtual seat design in improving the efficiency of content dissemination. As we discussed in Section 4.1, the surrounding seat arrangement helps the dissemination of streams with content sharing and distribution. To verify the advantage brought by the virtual seats, we simulates two Amphitheaters.



(a) Request rejection ratio **(b)** AQoS

Fig. 11. Effect of virtual seats.

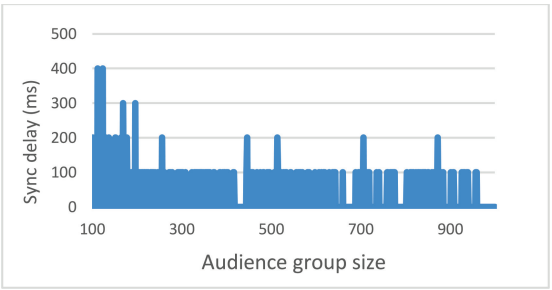


Fig. 12. Synchronization delay with gradually increasing audiences.

One with 500 audiences in the virtual seats, and one with zero audience. Other settings of this simulation are the same as the first part of the evaluation (Section 7.2).

Simulation Results. We plot the result in Figure 11, where the blue-diamond curve stands for the Amphitheater with audience sites and the red-square curve stands for the Amphitheater without audience sites. We can see from the figure that the performance is generally better when there are audience sites participating the session. Since the audience sites do not produce content, they play the role of hubs in the dissemination network. Recall that in our initial construction algorithm (Section 6.2), a performer site will turn to the audience sites to receive requested streams when the other performer sites are saturated. Thus, when there are audience sites in the session, this mechanism lowers the request rejection ratio of performer sites (and hence increases the AQoS) because more senders are provided to receive the stream from.

7.5. Synchronization Delay of User Churn

In this section, we evaluate the synchronization delay of a new audience when she joins. We simulate an Amphitheater with gradually increasing audiences from 100 to 1,000. The synchronization delay for a new audiences to join the theater is measured and plotted in Figure 12. The x-axis of the plot is the number of audiences in the system when the new audience joins. The y-axis is the synchronization delay.

Simulation Results. As we can see in the plots, in the early stage when the audience group size is not big enough, delays are longer and unstable (100 to 500 ms). This is due to the policy that new audience chooses its peers over the performers as sources when joining the amphitheater. In the early stage, the number of users is small and

hence there are fewer candidate sources to choose from. This incurs longer synchronization delay because a joining audience has a smaller chance to find a group of source audiences that hold in-synced contents. After the system gains more users ($>$300), the synchronization delay drops and stays stable under 200 ms. This shows the scalability of our Amphitheater on handling user churn.

8. DISCUSSION

Before we conclude this article, there are some omitted issues and potential features that we would like to discuss as some future directions.

Changing Seats during the Performance. Although contradicting the sense in the real world, allowing seat changing during the session may be an incentive for users to choose a virtual theater over a real one. However, a negative effect of this feature is the loss of the benefit brought by the virtual seat design. Like the heavy-tail distribution of channel popularity in IPTV, very likely the audience will flock to the same area in the auditorium for a more interesting viewpoint if we allow seat changing. The loss of even dispersion of the audiences may decrease the effectiveness of virtual seats on stream sharing.

Join and Leave of the Performer. An issue to be addressed next is the effect of intensive performer changes. In a complex stage performance, actors can come and go during a play. This introduces a huge change to the dissemination forest since performers are the roots of trees. Before we come up with an efficient solution for this performer churn, applications of the amphitheater system is restricted for simple stage performance where performers stay on stage throughout the play, for example, sport events, concerts, and lectures.

9. CONCLUSION

In this article, we present the 3DTI Amphitheater, a live broadcasting system for dissemination of 3DTI content. We identify the hierarchical priorities of streams and propose the concept of virtual seats which render the 3DTI environment manageable with more efficient content sharing. The Amphitheater framework is tested with real world network settings and configurations of real 3DTI system. This result shows that the Amphitheater outperforms prior 3DTI systems by boosting the AQoS while sustaining the same hundred-scale audience group.

REFERENCES

P. Agarwal, R. R. Toledano, Wanmin Wu, K. Nahrstedt, and A. Arefin. 2010. Bundle of streams: Concept and evaluation in distributed interactive multimedia environments. In *Proceedings of the IEEE International Symposium on Multimedia (ISM'10)*. 25–32.

A. Arefin, Zixia Huang, K. Nahrstedt, and P. Agarwal. 2012. 4D TeleCast: Towards large scale multi-site and multi-view dissemination of 3DTI contents. In *Proceedings of the 32th IEEE International Conference on Distribution Systems (ICDCS'12)*.

S. Chen, K. Nahrstedt, and I. Gupta. 2014. 3DTI amphitheater: A manageable 3DTI environment with hierarchical stream prioritization. In *Proceedings of the 5th ACM Multimedia Systems Conference (MMSys'14)*. 70–80.

S. Chen, P. Xia, and K. Nahrstedt. 2013. Impact of morphing-based frame synthesis on bandwidth optimization for 3DTI video. In *Proceedings of the 21th ACM International Conference on Multimedia (MM'13)*. 349–352.

G. Coulouris, J. Dollimore, and T. Kindberg. 2011. *Distributed Systems: Concepts and Design*. Addison-Wesley.

A. DeVincenzi, L. Yao, H. Ishii, and R. Raskar. 2011. Kinected conference: Augmenting video imaging with calibrated depth and audio. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CWCS'11)*. 621–624.

P. Th. Eugster, P. A. Felber, R. Guerraoui, and A. Kermarrec. 2003. The many faces of publish/subscribe. *ACM Comput. Surv.* 35, 2, 114–131.

D. Feldman and Y. Shavitt. 2007. An optimal median calculation algorithm for estimating Internet link delays from active measurements. In *Proceedings of the Workshop on End-to-End Monitoring Techniques and Services (E2EMON'07)* (Munich), 1–7.

M. Jain and C. Dovrolis. 2003. End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput. *IEEE/ACM Trans. Netw.* 11, 4, 537–549.

K. Nahrstedt, Z. Yang, W. Wu, A. Arefin, and R. Rivas. 2011. Next generation session management for 3D teleimmersive interactive environments. *J. Multimed. Tools Appl.* 51, 2, 593–623.

Netmap. 2014. www.caida.org/tools/visualization/mapnet.

PPLive. 2014. http://www.pptv.com/.

Teeve. 2009. http://cairo.cs.uiuc.edu/projects/teleimmersion/.

Ustream. 2014. http://www.ustream.tv/.

Z. Wang and J. Crowcroft. 1996. Quality-of-service routing for supporting multimedia applications. *IEEE J. Select. Areas Commun.* 14, 7, 1228–1234.

W. Wu, Z. Yang, I. Gupta, and K. Nahrstedt. 2008. Towards multi-site collaboration in 3D tele-immersive environments. In *Proceedings of the 28th IEEE International Conference on Distribution Systems (ICDCS'08)*, (Beijing, China), 647–654.

Zhenyu Yang, B. Yu, K. Nahrstedt, and R. Bajscy. 2006. A multi-stream adaptation framework for bandwidth management in 3D Tele-immersion. In *Proceedings of the International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'06)*. 14.

H. Yin, X. Liu, T. Zhan, V. Sekar, F. Qiu, C. Lin, H. Zhang, and B. Li. 2009. Design and deployment of a hybrid CDN-P2P system for live video streaming: experiences with LiveSky. In *Proceedings of the 17th ACM International Conference on Multimedia (MM'09)*. 25–34.

Youtube. 2014. http://www.youtube.com/.

Xinyan Zhang, Jiangchuan Liu, Bo Li, and Tak-shing Yum. 2005. CoolStreaming/DONet: A data driven overlay network for peer-to-peer live media streaming. In *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'05)*. 2102–2111.

Received May 2014; revised September 2014; accepted September 2014