

# A Case for 3D Streaming on Peer-to-Peer Networks

Shun-Yun Hu\*

Department of Computer Science and Information Engineering  
National Central University, Taiwan

## Abstract

One of the most serious issues holding back the widespread of 3D contents on Internet has been their inaccessibility due to large data volume. Many compression and progressive transmission techniques, as well as format standards, have been proposed in recent years to make 3D streaming increasingly viable for the efficient and accessible delivery of 3D contents. However, existing proposals have yet to seriously address one of the most important issues in practical adoption – a system’s *scalability* in terms of the number of concurrent users. We argue that due to 3D contents’ large data volume and interactive nature, client-server architecture, with its inherently fixed resource availability and high cost, will not be suitable to support popular Internet-scale 3D streaming. On the other hand, *peer-to-peer* (P2P) architectures hold the promise of both scalability and affordability. In this position paper, we describe the potential promises and challenges in adapting 3D streaming to P2P networks, using multi-user *networked virtual environment* (NVE) as an example. We also propose *Flowing LoD* (FLoD), a scalable, distributed and fault-tolerant P2P 3D streaming mechanism, that is based on *Voronoi-based Overlay Network* (VON), a P2P overlay specifically designed for NVE applications.

**CR Categories:** C.2.4 [Computer-Communication Networks]: Distributed Systems—Distributed applications; I.3.2 [Computer Graphics]: Graphics Systems—Distributed/network graphics; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual reality

**Keywords:** 3D streaming, peer-to-peer (P2P), networked virtual environment (NVE), scalability, Voronoi, overlay networks, visibility determination

## 1 Introduction

Since the beginning of the World Wide Web, people have dreamed about creating freely navigable 3D worlds. In fact, one of the key motivations behind drafting standards such as VRML was to create a 3D cyberspace [Pesce et al. 1994]. However, the vision has yet to be realized due to arguably the lack of three factors instrumental to the success of the Web: standardized data format and transmission protocol (i.e. HTML and HTTP), ease of content creation, and efficient content delivery (i.e. the client-server model). Although much efforts have been made on standardized 3D data formats (e.g. VRML and its successor X3D [Web3D 2006], MPEG4’s BIFS [ISO 1999], U3D [3DIF 2006], OpenHSF

[OpenHSF 2006]) and transmission protocols (e.g. VRTP [Brutzman et al. 1997], DWTP [Broll 1998]), 3D content creations remain in the hands of professional modelers and artists, while the sheer size of typical 3D data causes long download time, making them inaccessible to the average Internet users.

Audio and video data are also large in size, but thanks to *streaming* delivery, users need not wait for a complete download before using the contents. The possibility to stream audio and video has been crucial in their widespread popularity. Similarly, when broadband networks and 3D acceleration cards were adopted by average users in the late ’90s, streaming delivery of 3D data began to appear as a viability, raising hopes for the widespread adoption of 3D contents.

In 1996, Hoppe introduced the concept of *progressive meshes* [Hoppe 1996], which formed the basis to stream geometrical meshes from servers to clients, making interactions with 3D data possible without a complete download [Teler and Lischinski 2001]. Since then, much effort has been spent on various progressive transmission techniques, including the streaming of meshes, textures, animations, and scene graphs. Although many existing literatures use “progressive transmission” or “streaming of 3D contents” to more precisely describe this process, we anticipate the term “3D streaming” will be popularly adopted once 3D contents flow widely on the Internet. Therefore, in this paper we will use “3D streaming” to refer to all continuous real-time transmission of 3D contents (e.g. meshes, textures, animations, etc.) for the purpose to allow user interactions with contents as soon as possible.

However, existing techniques and systems have yet to address a crucial issue in 3D streaming’s adoptions – the *scalability* of the number of concurrent users of the system. To the best of our knowledge, almost all existing proposals use the *client-server* model for content delivery, where all users access and request 3D contents from a centralized server. Yet the very nature of 3D streaming necessarily places a heavy burden on the streaming server providing the service. Although large audio and video files are also delivered routinely on the Internet by client-server architectures, 3D streaming differs from video or audio streaming in that the selections of streaming contents require additional processing (e.g. for tasks such as visibility determination), which makes CPU provisioning also a serious issue in addition to bandwidth considerations. If scalability is achieved by adding more server-side resources (i.e. utilizing a *server-cluster*, where many servers located on a high-speed LAN are connected to the outside with large uplink bandwidth), new issues such as load balancing and fault tolerance are introduced [Chen et al. 2005], which in turn increases the design complexity and costs, severely limiting the number of potential adopters.

To make our discussions concrete, we will focus on 3D streaming in the context of multi-user *networked virtual environments* (NVEs), [Singhal and Zyda 1999] where users not only interact with objects, but also with each other in a potentially large-scale 3D environment. A good example of such NVE is *Massively Multiplayer Online Game* (MMOG) [Wikipedia 2006], which has been gaining growth and popularity rapidly as a major Internet application, and up to *hundreds of thousands* of users may interact simultaneously in visually stunning 3D worlds. MMOG may currently be the closest example to the original 3D cyberspace vision, so it can serve as a case study. A good question to ask is how MMOG may be adapted

\*e-mail: syhu@yahoo.com

Copyright © 2006 by the Association for Computing Machinery, Inc. Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions Dept, ACM Inc., fax +1 (212) 869-0481 or e-mail [permissions@acm.org](mailto:permissions@acm.org).

Web3D 2006, Columbia, Maryland, 18–21 April 2006.

© 2006 ACM 1-59593-336-0/06/0004 \$5.00

to domains beyond entertainment and become more accessible to ordinary users. Currently almost all MMOGs require users to pre-install the entire 3D contents used for game operation on their local machines, due to the large data volume that easily extends to over hundreds and even thousands of megabytes. Yet, we observe that each user needs to use, and is able to interact with, only small portions of the overall data at any given time. Therefore, if MMOG contents may be delivered through 3D streaming, we will be one step closer to the cyberspace vision. As 3D streaming is fundamentally processing and bandwidth intensive, the challenge to stream all 3D contents of a MMOG in real-time likely will not be met by existing client-server architectures cost-effectively.

In this position paper, we argue that the inherent large size and interactive nature of 3D contents make client-server architecture insufficient as a standard delivery model, while *peer-to-peer* (P2P) architectures hold the promises of being more scalable, affordable, and interactive. P2P has gained wide publicity and popularity through file-sharing applications in recent years, yet its central concept – a distributed network where participants contribute resources to support collective tasks – is applicable to a wider range of applications (e.g. *distributed hash table* [Stoica et al. 2001], *voice-over-IP* [Baset and Schulzrinne 2004], and *web cache* [Iyer et al. 2002]).

Existing file-sharing mechanisms such as *BitTorrent* [Cohen 2003] has demonstrated the feasibility to efficiently distribute large files to many users, with small server bandwidth use and quick client download time, by dividing a file into small pieces and utilizing the uplink bandwidth of other peers. Although currently, typical bandwidth on end-user networks is asymmetric (i.e. "uplink" bandwidth is smaller than "downlink" bandwidth), by concurrently downloading from multiple peers, good performance can still be achieved (e.g. average download between 240kbps to 500kbps have been reported in measurement studies [Izal et al. 2004; Pouwelse et al. 2005]). Suggestions have also been made that *BitTorrent*-style mechanisms may be adapted for streaming purposes [Wu and Chiueh 2006]. As broadband adoptions continue, the infrastructure is also becoming ever suitable for P2P-based content distributions.

The basis of our proposal to use P2P for 3D streaming is the emerging research of *P2P-based networked virtual environments* (P2P-NVE) [Hu et al. 2006], where the workload and maintenance of a large NVE is distributed to all participating computers, such that the collective bandwidth and processing power are utilized to support user interactions and data processing. We propose that if 3D streaming were to be widely adopted by many concurrent users, mechanisms in P2P-NVE may provide the necessary support to feasibly realize scalable and efficient 3D streaming.

The contributions of this paper are the problem formulation of 3D streaming for NVE-like applications on P2P networks, a brief survey of existing 3D streaming techniques, and the proposal of *Flowing LoD* (or *FLoD*), a P2P-based 3D streaming mechanism. The rest of the paper is organized as follows. We first give a background of related research in Section 2. In Section 3, a generalized model of the problem is given, followed by an explanation of the promises and challenges of using P2P for 3D streaming. We describe in Section 4 the tentative design of FLoD, discuss its merits and issues in Section 5, and conclude this paper in Section 6.

## 2 Background

We first provide some background related to the main theme of this paper, including: 3D streaming, the emerging study of P2P-based networked virtual environments (P2P-NVE), and media streaming (i.e. audio and video) on P2P networks.

### 2.1 3D Streaming

In general, the main goal of 3D streaming is to provide 3D contents in real-time for users over a network connection, such that the interactivity and visual qualities of contents may match as closely as if they were stored locally. The main resource bottleneck is usually assumed to be the bandwidth and not rendering or processing power of the clients [Teler and Lischinski 2001]. To achieve this goal, simplification and progressive transmission of the contents are two dominant strategies [Rusinkiewicz and Levoy 2001].

Existing 3D streaming techniques may be categorized into four main types. The first is the streaming of a single 3D object. *Progressive meshes* [Hoppe 1996] is a method to store an arbitrary triangular mesh as an appearance-preserving but much coarser *base mesh* and a number of refinement pieces. A remote user may immediately start to view or interact with the object once the base mesh is downloaded. Further streaming of additional pieces will incrementally refine the base mesh and restore the original mesh exactly. Progressive meshes was the basis that sparked much of the subsequent 3D streaming research (e.g. compressed progressive meshes [Pajarola and Rossignac 2000], streaming over lossy transmission links [Chen et al. 2003], over wireless channels [Yang and Kuo 2003], and QoS-related streaming [Chen and Nishita 2002]). As the original progressive meshes did not consider the user's viewing angle, yet visual relevance may be improved if current visible areas are given priorities during transmission, *view-dependent* streaming has since been investigated [Kim et al. 2004]. For streaming high resolution models, a method based on the non-polygonal point rendering system *QSplat*, was also proposed [Rusinkiewicz and Levoy 2001]. Object streaming has also been studied in the context of specific file formats such as X3D [Fogel et al. 2001] and MPEG-4's BIFS [Hosseini and Georganas 2002]. As typical 3D contents may include other data types besides polygonal meshes, streaming techniques may also tilt toward specific types of data, such as textures [Marvie and Bouatouch 2003], animations [Hijiri et al. 2000], and scene descriptions [Sahm et al. 2004], among others.

The second type of 3D streaming extends the delivery of a single object to that of an entire scene. In *scene streaming*, a collection of objects are placed at arbitrary positions within a virtual environment, and are streamed to clients according to visibility or user interest (along with object placements information). Scene streaming usually aims to provide a *remote walkthrough* (i.e. navigation) or multi-user NVE experience. Due to its more complicated nature, a wider range of strategies are needed to reduce bandwidth use while achieving good interactivity and visual quality. As many more objects may exist than what the user can see at a given time, scene streaming is generally divided into two stages: *object selection* and *object transmission*. For the first stage, visibility determination techniques are often used at the server to cull away objects invisible to the user. Once some objects are selected for transmission, streaming techniques for single objects are then employed. Many techniques useful for scene streaming were first described by [Schmalstieg and Gervautz 1996], where each user's visibility is limited to a circular *area of interest* (AOI). A server determines and transmits the set of visible objects at different *level of detail* (LOD) of model geometries to clients. Clients also *prefetch* objects so that the wait for downloading may be unnoticeable to users. A subsequent work replaced the use of discrete LODs with continuous (*smooth*) LODs and named this process *remote rendering* [Hesina and Schmalstieg 1998]. In [Teler and Lischinski 2001], pre-rendered image-based *impostors* served as the lowest LOD of 3D objects to allow faster initial visualization. An *online optimization algorithm* was also used by the server to provide the best perceived visual quality under a fixed bandwidth budget, thereby enable immediate and practical navigation under even very

small bandwidth. *Cyberwalk* [Chim et al. 2003] utilizes progressive meshes for transmissions, and focuses on caching and prefetching techniques to enhance visual perceptions while reducing the *response time* between initial queries and obtainment of objects. Some commercial MMOG systems use scene streaming to support dynamic contents (e.g. ActiveWorlds [AW 2006], Second Life [LindenLab 2006], and There.com [ForterraSystems 2006]), but few public information is available on their mechanisms.

A third type of 3D streaming has been applied to scientific visualizations. Certain scientific computing may generate vast amounts of spatial data that require visualization in 3D space for analysis or comprehension. Streaming for these 3D data differs from object-based streaming in that the data volume is usually much larger, and may involve time-dependent deformations of the 3D models that require a complete refresh (i.e. re-download) of a given 3D dataset (in contrast to animating a 3D object where only animation data or simple events need to be sent). Accuracy of model representation is also given priority over visual aesthetics. Olbrich and Pralle pioneered a series of such 3D streaming systems for scientific visualizations based on VRML and a customized *DocShow-VR* (DVR) format [Olbrich and Pralle 1999]. Another project *ViSTA* focuses on the interactive visualizations of *computational fluid dynamics* (CFD). It utilizes server-cluster for the parallel generations and post-processing of raw data to allow view-dependent visualizations [Gerndt et al. 2004]. However, the lab environment for these scenarios, where large data volumes are transferred and processed by high performance networks and graphics servers, is not yet transferable to Internet environment where bandwidth is often limited.

The fourth type of 3D streaming may be called *image-based streaming*, where 3D contents are stored at the server and not transmitted to clients. Client machines instead receive 2D rendered images generated in real-time by the server [Cheng et al. 2004]. This approach is suitable (and even necessary) when the client has only thin functionalities (i.e. low processing power with no 3D acceleration capability) and has the benefit of using only constant bandwidth. However, the severe processing requirements on the server may bring poor scalability and interactivity.

Our focus in this paper is on scene streaming as we seek to use 3D streaming to support Internet-scale NVEs. A generalized model of 3D streaming for NVE will be described in Section 3.

## 2.2 P2P-based Networked Virtual Environments

While the research on 3D streaming is relatively new, the study of multiuser *networked virtual environments* (NVE), where people at different physical locations may interact in the same virtual world, has seen a longer history dating back to U.S. Department of Defense's *SIMNET* in the 1980's [Singhal and Zyda 1999]. The network model of NVE has since evolved from the least scalable *point-to-point* (i.e. all participating nodes send messages to each other), to the later *client-server* (i.e. a centralized server receives, processes, and relays messages for all nodes) and today's *server-cluster* model as in commercial MMOGs [Hu et al. 2006]. However, server-based architectures likely will not meet the challenge to scale the number of concurrent users to the next order of magnitude (i.e. in the range of over a million), as server-side resources are usually fixed and require prior provisioning, yet the required resources likely will be prohibitively costly and complicated to maintain.

Significant efforts in NVE research have been on using *multicast* to provide better scalability without the limitations of centralized servers. Multicast (also known as *IP multicast*) allows a sender to notify multiple recipients with only a single transmission. By dividing the NVE into various regions (each assigned a unique mul-

ticast address), users can then communicate only with neighboring users in the same region via multicast, significantly reducing the total amount of messages being sent and processed (e.g. NPSNET [Macedonia et al. 1995], Spline [Barrus et al. 1996], and DWTP [Broll 1998]). Unfortunately, the very nature of multicast requires substantial router-level support, which makes its deployment non-trivial and still unavailable on much of the Internet. The number of available multicast addresses is also limited, which makes supporting concurrent NVEs with many participants unfeasible. As a result, *application-layer multicast* (ALM) [Liebeherr et al. 2002] (or *end-system multicast* [Chu et al. 2002]) have been proposed in recent years to provide multicast capability with overlay networks, where messages reach their destinations through node relays. However, the relay increases transmission latency, limiting its applicability for NVE systems (which generally require responsiveness). Another common problem with both IP multicast and ALM is that these approaches require the NVE be partitioned into fixed-size regions, which makes determining the proper region size a difficult task for message scoping to be efficient [Hu and Liao 2004].

Recently, peer-to-peer (P2P) solutions that seek to address the drawbacks of both server and multicast-based approaches have been proposed, with the promise of being more scalable, affordable, and easy to deploy [Keller and Simon 2003; Knutsson et al. 2004; Hu and Liao 2004]. As each node's bandwidth is limited, the premise of P2P-based NVE is that through connections with only a few neighbors, the *topology* of all nodes may be maintained and kept fully-connected (i.e. a path exists between any two nodes on the overlay. Note that "neighbors" here refer to proximity in the NVE, not physical routing hops). At the same time, each node can receive the relevant messages generated by neighbors within its *area of interest* (AOI). The central challenge in designing P2P-based NVE system thus is to devise a mechanism where peer nodes may discover each other correctly and efficiently (i.e. solving a *neighbor discovery problem*). Aside from some differences in the correctness and efficiency of topology maintenance, most current P2P-NVE proposals can provide the following simple function without server involvement: given a node's coordinate in the NVE and its AOI (usually specified as the radius of a circle, centered at the node), return the information of neighboring nodes within this AOI (i.e. its *AOI neighbors*). The information should include the AOI neighbors' coordinates in the NVE and IP addresses for contact. This function will be relevant to *FLoD*'s design, as discussed later in Section 4.

## 2.3 Media Streaming on P2P Networks

To improve a system's scalability without overloading the server, many studies have been done on how to adapt audio or video streaming to a P2P environment [Xu et al. 2002; Tran et al. 2003]. Various approaches usually construct *application-layer multicast* trees rooted at the server, with clients as nodes in the tree. Issues such as tree depth, link-degree of each node, and tree restructure in case of node departures thus are the main issues. Although similarities exist between 3D streaming and other media streaming (e.g. data stream is usable before download is complete, sequential transfer of data blocks, and the applicability of prefetching techniques), 3D scene streaming differs from other media streaming in one fundamental aspect: contents of 3D streaming is based on the results of visibility determination, as each user may have different visibility, individual data streams are mostly unique unless users are in close proximities within the NVE. In other words, *3D streaming works as if every individual user is watching a different movie, from potentially unlimited number of selections* (i.e. no two movies are alike). Due to this major difference, existing P2P media streaming techniques may not be straightforwardly used for 3D streaming.

### 3 3D Streaming for NVE on P2P Networks

#### 3.1 System Model and Assumptions

We assume that a number of 3D objects of various sizes and shapes are in the NVE, each has a position and orientation in a 3D space. Objects are completely definable by polygonal meshes and the associated textures (currently, we do not consider other data types such as animations or colors). There may also be a number of *avatars*, which are 3D representations of participating users. Each user navigates the NVE through a client program (thus we will use the terms *user*, *node*, and *client* interchangeably). As there could potentially be many objects and avatars, it would not be feasible nor necessary to see and interact with all of them. Each user's visibility and interaction thus is limited to a circular *area of interest* (AOI) centered at the user's current location (Figure 1). For simplicity, we do not yet consider animations of the avatars and treat them just as other 3D objects (except that avatars may move *dynamically* while other objects always remain *statically* at fixed locations).

For a given 3D object, we assume that its mesh and texture can be decomposed into a *base piece* and many *refinement pieces* (Figure 2). The specific decomposition is beyond the scope of this paper, but whichever the mechanism, we assume that the user is provided with a minimal working set of 3D objects once the base pieces are obtained, such that the scene can be rendered and navigation may start. Progressive meshes [Hoppe 1996] and related techniques may be used for mesh decompositions, while progressive encodings of image formats such as GIF, JPEG, or PNG, may be used for texture decompositions [Marvie and Bouatouch 2003].

All 3D contents are initially stored at a server, and clients obtain them through a streaming process from either the server or other clients (also referred as *peers*). Once an initial set of base pieces is obtained, rendering and navigation may begin at the client.

#### 3.2 Requirements for P2P-based 3D Streaming

From the user's perspective, the most important qualities of a 3D streaming mechanism are *speed* and *visual quality*. As the latter is beyond our scope, we will focus on speed, which can be more specifically divided into *startup time* (i.e. the time from connecting to a server to starting navigation) and *completion time* (i.e. the time to completely download a 3D object). Ideally, 3D streaming would work much like the existing Web, where the display (and user interaction) can happen within *seconds* after an URL is entered. Thus, 3D streaming mechanisms may be judged by how well they minimize the startup and completion time.

From the server's perspective, it is preferable if repetitive requests of the same contents can be minimized and redirected to other clients that could serve the requests. As mentioned before, scene streaming has two stages: the selection of 3D objects (as determined by the requester's visibility) and the efficient transmission of the selected objects. This first stage involves some kind of visibility determination, while the second stage may involve calculating an optimal transmission strategy (e.g. the *online optimization algorithm* in [Teler and Lischinski 2001]). It is desirable for the server to minimize its involvement in these tasks, as they will cost processing power for each client served. Ideally, if visible object selection (i.e. visibility determination) is done by clients, then server-side processing can be much conserved for serving data requests only.

3D streaming is similar to audio and video streaming in that contents are downloaded and used in a sequential manner, due to the sequential nature of progressive meshes (or textures) reconstruction.

Although download may be speeded up by obtaining pieces from multiple sources concurrently, the pieces still need to be processed in a more or less sequential manner.

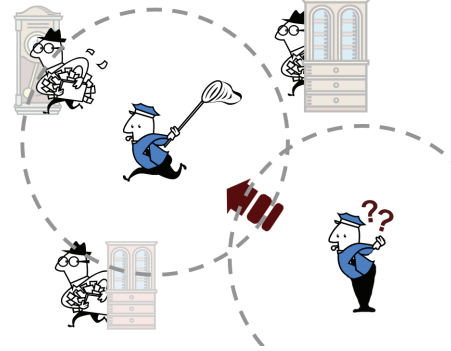


Figure 1: Example of a NVE with *objects* (clock and closets) and *avatars* (cop and thieves). Circle is the *area of interest* (AOI). Note how a neighboring avatar becomes visible after the AOI has moved.

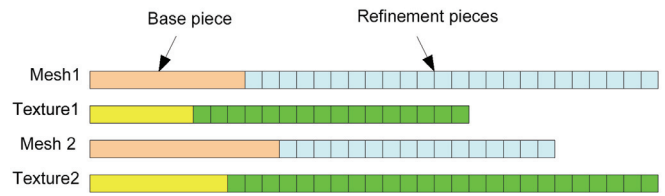


Figure 2: 3D content decompositions.

#### 3.3 Challenges

To meet these requirements, we would like to use client resources for *object selections* and *transmissions*, introducing these issues:

##### Distributed Visibility Determination

Although it is desirable if visibility determination can be done without server involvement, only the server possesses complete knowledge of scene descriptions (i.e. object placements within the NVE). Scene descriptions thus need to be distributed to peers efficiently before clients can do visibility determination distributively.

##### Peer and piece selection

The central goal in 3D streaming is to minimize overall download time and optimize the visual qualities for a given time or bandwidth budget. This involves selecting the proper peers (i.e. *peer selection*) and requesting for the proper data pieces (i.e. *piece selection*). As relevant pieces of a 3D object may reside on different peers, finding the right peers can be an issue, where both content availability and network conditions need to be considered. Additionally, once a peer is contacted, the question of which pieces should be obtained needs to be considered. For a given object, we also need to download the pieces in a more or less sequential manner.

##### Peer heterogeneity

In a real network environment, each client's resource capacities likely will differ greatly and might fluctuate. Besides unpredictability in the clients' resource capacities, client availabilities could also fluctuate due to network congestion or node failures. In order to fully utilize client resources, differentiated and adaptive use of these resources is likely required.

## 4 Design of FLoD

### 4.1 Overview

In this section we will outline our proposed mechanism for 3D streaming on P2P network, called *Flowing LoD (FLoD)*. We choose to use *Voronoi-based Overlay Network (VON)* as the underlying P2P layer, as it has demonstrated scalability, efficiency, and reliability [Hu et al. 2006]. VON provides a simple function: given a coordinate and a visibility radius (i.e. the *AOI radius*), it returns the information of the current AOI neighbors. As a node moves around, it sends VON the new coordinates to obtain an updated list of AOI neighbors. By using VON, we can easily learn of relevant nodes to interact with. Note that other P2P-NVE designs may also be used, as long as they provide the correct information on AOI neighbors.

The main design rationale behind FLoD is that because nodes often share visible areas with their AOI neighbors, it is thus likely that the neighbors already possess some relevant 3D contents. By requesting data from them first, not only server will be relieved from serving the same content repetitively, download may even be faster than when the server is congested, as multiple data sources are available.

In FLoD's design, each 3D object has a unique ID and *location point* (i.e. a point coordinate representative of the object) that is specified in a *scene description* along with its *orientation* and *scale* (currently, the scene description contains only the id, location point, orientation and scale of objects). To distribute scene descriptions to clients efficiently, the entire NVE is divided into fixed-size square *cells*, each of which contains a small scene description specifying object placements within the cell (note that other shapes such as triangle / rectangle / hexagon may also be used for the partition, yet performance difference will be trivial as the partitioning is not for scoping message exchange). The first stage in 3D streaming – the selection of 3D contents, can thus be performed in a fully-distributed manner, as each client can locally and independently determine which cells its AOI currently covers (Figure 3). The client then requests these scene descriptions from either other clients or the server (in case no AOI neighbor answers the request). Once the scene descriptions are obtained, the clients would then obtain the 3D objects using techniques for single object streaming, according to certain priority criteria (determined individually by user preference). A view is rendered with progressively refined details as the relevant objects are streamed from either other peers or the server (which acts as the final fallback for data requests).

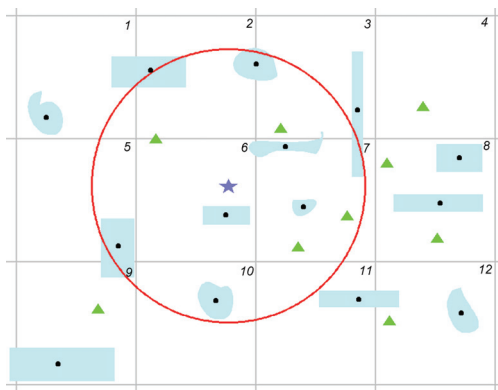


Figure 3: Schematic of a NVE divided into *cells*. Big circle is the AOI of the star node while triangles are other *user nodes*. Various shapes are 3D objects, with their *location points* as dots. Note that cell ids can be calculated from world dimensions and cell size alone.

### 4.2 Procedures

We now describe in more details FLoD's main procedures:

#### Login

1. The joining node enters the P2P network by specifying a join location and AOI-radius to VON, which then contacts the *gateway server* to obtain a unique ID. VON also finds and reports an initial list of AOI neighbors through its distributed query mechanism.
2. An overall description of the NVE (e.g. its dimensions and cell size) is also obtained from the *gateway server*.
3. Joining node initiates *Obtain Scene Description*.

#### Obtain Scene Description

1. The requesting node calculates which cells does its AOI cover, and utilizes the *Request for Piece* procedure (with *cell id* as the parameter) to obtain the *scene descriptions* for these cells.
2. Once the *scene descriptions* are obtained, the requesting node obtains missing 3D objects not available in its local cache using the *Obtain 3D Objects* procedure.

#### Obtain 3D Objects

1. Missing pieces (i.e. *base piece* or *refinement pieces* of mesh or textures) for each 3D object to be obtained are first determined.
2. For each missing data piece, *Request for Piece* procedure is invoked (with *object id* and *piece id* as the parameters). Refinement pieces are obtained in a mostly sequential fashion.
3. Once the data piece is obtained, 3D objects are rendered with currently available data pieces according to their specified locations, orientations, and scales in the *scene description*.

#### Request for Piece

1. The requesting node prepares a list of potential *data source nodes* composed of currently connected neighbors (mostly its AOI neighbors) and the *gateway server*. The list is sorted by certain priority criteria (e.g. latency, bandwidth, or data availability).
2. The requesting node sends out requests to each node on the list in a roundrobin (i.e. sequential) fashion.
3. Nodes that receive the requests may either *respond* to the request by sending back the data piece, or *deny* the request due to unavailability of the data piece or bandwidth.
4. If a request is denied, requests will be sent again to other *data source nodes*. As the *gateway server* is part of the pool, all data requests may eventually be fulfilled by the server.

#### Move

1. A node moves by sending position update to VON.
2. If new neighbors are discovered via VON, their avatar representations are requested by the *Obtain 3D Objects* procedure.
3. If the node has entered any new cells which it does not have the scene descriptions, *Obtain Scene Description* is invoked.

#### Logout

1. A node may simply disconnect from the P2P network, as the system is basically fully-distributed. Failure or departure of any single user node will not affect the system operation.
2. Other nodes will learn about the departing node through updated neighbor list obtained from VON.

## 5 Evaluation

### 5.1 Discussions

We will discuss FLoD's properties according to the issues described previously in Section 3.3, as well as some of its limitations.

#### Distributed Visibility Determination

Due to the pre-partitioning of the NVE into cells and our design to limit a node's visibility to be within its AOI, visibility determination thus can be done in a completely distributed fashion without the need for server intervention. This property is important for scalability as there is no processing cost for visibility determination on the part of the server for each additional node joined.

#### Peer and piece selection

The challenge of finding the peers with relevant content is solved by sending requests to *AOI neighbors*, while the issue of deciding which pieces to request is dealt by the round-robin requests from a pool of potential *data source nodes* (composed of AOI neighbors and the server). As there are a number of *data source nodes* from which data pieces can be concurrently obtained, the streaming process may be faster than if a single data source is available. Although it is possible that sometimes no neighbors exist within a node's AOI, as VON's design still maintains connections with a minimal set of nearby nodes [Hu et al. 2006], data requests may still be fulfilled by peer nodes without having to resort to the server.

#### Peer heterogeneity

The parallelism in FLoD's streaming process increases the tolerance for potential node failures or data unavailability, as the requesting node may simply send request to the next available node in its pool of potential data sources. As no single node (except the gateway server) bears more responsibility than any other, failure of any single user node will not affect the system's overall operation.

#### Limitations

In the current design, each node retrieves data pieces from only their AOI neighbors, which might be too limiting a subset of qualified nodes. Efficiency at matching data request and supply thus might not be optimal. We have not yet considered caching or prefetching techniques in a P2P environment, however, they are essential for any streaming to be effective. For P2P-based retrieval to work, sufficiently large number of peers must be within the AOI of the requesting node, otherwise it will fall back to server-based retrieval modes. Finally, efficient representations of 3D contents are still under active research, where the progress will much impact the applicability of any content delivery mechanisms. One particular issue is the size of textures, which often is orders of magnitude larger than that of geometrical models in MMOG-like scenarios. However, new research directions such as *texture synthesis* [Efros and Leung 1999] may provide some interesting solutions.

### 5.2 Implementation

We are currently in the process of implementing and evaluating FLoD. As FLoD uses Voronoi-based Overlay Network for neighbor discovery, the implementation will use the cross-platform open source library *VAST* [VAST 2006]. FLoD is also being developed within the research project *ASCEND (Adaptive Scalable Cooperative Environment for NVE Developments)* [ASCEND 2006], which seeks to investigate the major issues in adapting large-scale NVEs to the P2P environment.

## 6 Conclusion

Although efforts to create a 3D cyberspace date back to the early days of the Web, 3D contents have yet to find a way to be widely used by ordinary Internet users. However, improvements in network infrastructure and steady progress in format standards are now bringing new hopes to the original vision. While the ease of content creations remains a challenge, *3D streaming* may effectively address the delivery problem. We suggest that although various streaming methods have been devised in recent years, their processing and bandwidth intensive nature likely will make servicing many concurrent users impractical or prohibitively expensive by client-server architectures. A promising alternative is to use *peer-to-peer* networks, and specifically, to exploit the *neighbor discovery* mechanism of P2P-NVE systems in solving issues of distributed visibility determination, peer and piece selection, and peer heterogeneity. We have also described a P2P-based 3D streaming design for NVE applications called *FLoD*, of which we will continue to work on its implementation and evaluations. Our proposal is only the first of many, and we expect more will come. 3D streaming on P2P networks is an important topic worthy of the attentions of both the graphics and networking communities. By identifying the basic issues, we hope to generate interests in this promising direction to realize more convenient access of 3D contents. The impacts of P2P have been demonstrated in many areas, it is perhaps time to see it be applied in realizing the original vision of the *3D cyberspace*.

## 7 Acknowledgments

This work was supported by NSC 94-2213-E-008-001. I wish to thank Prof. Jehn-Ruey Jiang for his supports, Ye-Zen Chang Chen for valuable discussions, and Chen-Yu Yeh (Yakko) for Figure 1.

## References

- 3DIF, 2006. 3d industrial forum. <http://www.3dif.org/>.
- ASCEND, 2006. Ascend project. <http://ascend.sourceforge.net>.
- AW, 2006. Activeworlds. <http://www.activeworlds.com/>.
- BARRUS, J. W., WATERS, R. C., AND ANDERSON, D. B. 1996. Locales: Supporting large multiuser virtual environments. *IEEE Comput. Graph. Appl.* 16, 6, 50–57.
- BASET, S. A., AND SCHULZRINNE, H., 2004. An analysis of the skype peer-to-peer internet telephony protocol.
- BROLL, W. 1998. Dwtp – an internet protocol for shared virtual environments. In *Proc. Symp. VRML*, 49–56.
- BRUTZMAN, D., ZYDA, M., WATSEN, K., AND MACEDONIA, M. 1997. virtual reality transfer protocol (vrtp) design rationale. In *Proc. WET-ICE*.
- CHEN, B.-Y., AND NISHITA, T. 2002. Multiresolution streaming mesh with shape preserving and qos-like controlling. In *Proc. ACM Web3D*, 35–42.
- CHEN, Z., BODENHEIMER, B., AND BARNES, J. F. 2003. Robust transmission of 3d geometry over lossy networks. In *Proc. Intl. Conf. on 3D Web Technology*, 161–ff.
- CHEN, J., WU, B., DELAP, M., KNUTSSON, B., LU, H., AND AMZA, C. 2005. Locality aware dynamic load management for massively multiplayer games. In *Proc. ACM PPOPP*, 289–300.

- CHENG, L., BHUSHAN, A., PAJAROLA, R., AND ZARKI, M. E. 2004. Real-time 3d graphics streaming using mpeg-4. In *Proc. IEEE/ACM Wksp. on Broadband Wireless Services and Appl.*
- CHIM, J., LAU, R. W. H., LEONG, H. V., AND SI, A. 2003. Cyberwalk: A web-based distributed virtual walkthrough environment. *IEEE Trans. on Multimedia* 5, 4, 503–515.
- CHU, Y. H., RAO, S. G., SESHAN, S., AND ZHANG, H. 2002. A case for end system multicast. *IEEE JSAC* 20, 8, 1456–1471.
- COHEN, B. 2003. Incentives build robustness in bittorrent. In *Proc. Wksp. on Economics of Peer-to-Peer Systems*.
- EFROS, A. A., AND LEUNG, T. K. 1999. Texture synthesis by non-parametric sampling. In *Proc. ICCV*, 1033–1038.
- FOGEL, E., COHEN-OR, D., IRONI, R., AND ZVI, T. 2001. A web architecture for progressive delivery of 3d content. In *Proc. ACM Web3D*, 35–41.
- FORTERRASYSTEMS, 2006. There.com. <http://www.there.com>.
- GERNDT, A., HENTSCHEL, B., WOLTER, M., KUHLEN, T., AND BISCHOF, C. 2004. Viracocha: An efficient parallelization framework for large-scale cfd post-processing in virtual environments. In *Proc. SC2004*.
- HESINA, G., AND SCHMALSTIEG, D. 1998. A network architecture for remote rendering. In *Proc. Intl. Wksp. DIS-RT*, 88.
- HIJIRI, T., NISHITANI, K., CORNISH, T., NAKA, T., AND ASAHARA, S. 2000. A spatial hierarchical compression method for 3d streaming animation. In *Proc. ACM VRML*, 95–101.
- HOPPE, H. 1996. Progressive meshes. *Computer Graphics* 30, Annual Conference Series, 99–108.
- HOSSEINI, M., AND GEORGANAS, N. D. 2002. Mpeg-4 bifs streaming of large virtual environments and their animation on the web. In *Proc. ACM Web3D*, 19–25.
- HU, S.-Y., AND LIAO, G.-M. 2004. Scalable peer-to-peer networked virtual environment. In *Proc. ACM SIGCOMM 2004 wksp. on NetGames '04*, 129–133.
- HU, S.-Y., CHEN, J.-F., AND CHEN, T.-H. 2006. Von: A scalable peer-to-peer network for virtual environments. *IEEE Network (accepted)*, <http://vast.sf.net/docs/pub/2006-hu-VON.pdf>.
- ISO/IEC 14496-1. 1999. *Information Technology – Coding of audiovisual objects, Part 1: Systems*, January.
- IYER, S., ROWSTRON, A., AND DRUSCHEL, P. 2002. Squirrel: A decentralized peer-to-peer web cache. In *Proc. 21st ACM SIGACT-SIGOPS Symp. on Principles of Dist. Comp. (PODC)*.
- IZAL, M., URVOY-KELLER, G., BIRSACK, E., FELBER, P., HAMRA, A. A., AND GARCES-ERICE, L. 2004. Dissecting bittorrent: Five months in a torrents lifetime. In *Proc. Passive and Active Network Measurement (PAM 2004)*, 1–11.
- KELLER, J., AND SIMON, G. 2003. Solipsis: A massively multi-participant virtual world. In *Proc. PDPTA 03*, 262–268.
- KIM, J., LEE, S., AND KOBELT, L. 2004. View-dependent streaming of progressive meshes. In *Proc. SMI'04*, 209–220.
- KNUTSSON, B., LU, H., XU, W., AND HOPKINS, B. 2004. Peer-to-peer support for massively multiplayer games. In *Proc. IEEE INFOCOM*, 96–107.
- LIEBEHERR, J., NAHAS, M., AND SI, W. 2002. Application-layer multicasting with delaunay triangulation overlays. *IEEE J. Sel. Areas Commun. (JSAC)* 20, 8, 1472–1488.
- LINDENLAB, 2006. Second life. <http://secondlife.com/>.
- MACEDONIA, M., ZYDA, M., PRATT, D., BRUTZMAN, D., AND BARHAM, P. 1995. Exploiting reality with multicast groups. *IEEE Comput. Graph. Appl.* 15, 5, 38–45.
- MARVIE, J.-E., AND BOUATOUCH, K. 2003. Remote rendering of massively textured 3d scenes through progressive texture maps. In *Proc. 3rd IASTED Conf. VIIP*, vol. 2, 756–761.
- OLBRICH, S., AND PRALLE, H. 1999. Virtual reality movies - real-time streaming of 3d objects. *Computer Networks* 31, 21, 2215–2225.
- OPENHSF, 2006. Openshf initiative. <http://www.openshf.org/>.
- PAJAROLA, R., AND ROSSIGNAC, J. 2000. Compressed progressive meshes. *IEEE Trans. Vis. Comput. Graph.* 6, 1, 79–93.
- PESCE, M., KENNARD, P., AND PARISI, A. 1994. Cyberspace. In *Proc. First Intl. Conf. on the World Wide Web*.
- POUWELSE, J., GARBACKI, P., EPEMA, D., AND SIPS, H. 2005. The bittorrent p2p file-sharing system: Measurements and analysis. In *Proc. 4th Intl. Wksp. on Peer-to-Peer Systems (IPTPS'05)*.
- RUSINKIEWICZ, S., AND LEVOY, M. 2001. Streaming qsplat: A viewer for networked visualization of large, dense models. In *Proc. Symp. Interactive 3D Graphics*, 63–69.
- SAHM, J., SOETEBIER, I., AND BIRTHELMER, H. 2004. Efficient representation and streaming of 3d scenes. *Computers & Graphics* 28, 1, 15–24.
- SCHMALSTIEG, D., AND GERVAUTZ, M. 1996. Demand-driven geometry transmission for distributed virtual environments. *Computer Graphics Forum* 15, 3, 421–433.
- SINGHAL, S., AND ZYDA, M. 1999. *Networked Virtual Environments: Design and Implementation*. ACM Press.
- STOICA, I., MORRIS, R., KARGER, D., KAASHOEK, F., AND BALAKRISHNAN, H. 2001. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. ACM SIGCOMM*, 149–160.
- TELER, E., AND LISCHINSKI, D. 2001. Streaming of complex 3d scenes for remote walkthroughs. *EUROGRAPHICS* 20, 3.
- TRAN, D., HUA, K., AND DO, T. 2003. Zigzag: An efficient peer-to-peer scheme for media streaming. In *Proc. IEEE INFOCOM 2003*, 1283–1292.
- VAST, 2006. Vast project. <http://vast.sourceforge.net>.
- WEB3D, 2006. Web3d consortium. <http://www.web3d.org>.
- WIKIPEDIA, 2006. <http://en.wikipedia.org/wiki/MMOG>.
- WU, G., AND CHIUEH, T.-C. 2006. How efficient is bittorrent? In *Proc. 13th Annual Multimedia Comp. and Netw. (MMCN'06)*.
- XU, D., HEFEEDA, M., HAMBRUSCH, S., AND BHARGAVA, B. 2002. On peer-to-peer media streaming. In *Proc. IEEE Conf. on Distributed Computing and Systems*, 363–371.
- YANG, S., AND KUO, C.-C. J. 2003. Robust graphics streaming in walkthrough virtual environments via wireless channels. In *Proc. IEEE Globecom 2003*.

