



# Peer-to-Peer 3D Streaming

Virtual worlds have become very popular in recent years, with trends toward larger worlds and more user-generated content. The growth of 3D content in virtual worlds will make real-time content streaming (or 3D streaming) increasingly attractive for developers. To meet the demands of a large user base while lowering costs, peer-to-peer (P2P) content delivery holds the promise for a paradigm shift in how future virtual worlds will be deployed and used. The authors define both the problem and solution spaces for P2P 3D streaming — by outlining its requirements and challenges — and categorize existing proposals.

**Shun-Yun Hu  
and Jehn-Ruey Jiang**  
*National Central University,  
Taiwan*

**Bing-Yu Chen**  
*National Taiwan University*

**M**ultiuser 3D virtual environments (VEs) such as massively multi-player online games (MMOGs) have become very popular in recent years. World of Warcraft, a role-playing MMOG in which players participate in epic battles and adventures between two opposing camps, had more than 11 million paying subscribers in 2008. Second Life,<sup>1</sup> a social MMOG entirely built by its “residents,” also boasts more than 1.4 million active accounts and nearly US\$9 million worth of virtual item transactions each month. Due to the demand for more realistic worlds, VEs have become larger and more dynamic. As content becomes easier to create and cheaper to host, more developers — even individuals — are build-

ing virtual worlds (for example, Second Life hosted 34 Tbytes of user-generated content in 2007). However, the trends toward larger worlds and larger numbers of worlds are beginning to reveal the inadequacy of current VE installation methods, which require users to preinstall the full content via DVDs or a prior download. Easier access to this various and massive amount of VE content thus demands 3D streaming for content distribution.<sup>2,3</sup>

3D streaming delivers 3D content over networks in real time to let users navigate a VE without a complete content installation. Similar to audio or video media streaming, 3D streaming requires that the content be fragmented into pieces before it can be transmitted,

reconstructed, and displayed. However, unlike linearly streamed audio or video, 3D streaming is highly interactive and nonlinear in nature, with the streaming sequence based on each user's unique visibility or interest area (that is, the landscape or objects the user can see, or is interested in seeing, within the virtual world).

Current 3D streaming schemes adopt the client-server (C/S) model for content delivery. However, such architecture is difficult to scale because 3D streaming is both data- and processing-intensive. Prohibitively vast amounts of server-side bandwidth and CPU power are required for a massive audience. On the other hand, peer-to-peer (P2P) networks offer highly scalable yet affordable computing and content-sharing capabilities. Given that users in a 3D scene could own similar content due to overlapping visibility, they can obtain content from each other in a P2P manner. However, although P2P media streaming has progressed significantly in recent years, it might not be directly applicable to 3D content due to different content access patterns. In both live and on-demand media streaming, content is often sent linearly after a starting point, whereas access to 3D content is rather arbitrary and nonlinear, and depends much on real-time user behaviors.<sup>4</sup> New insights and novel designs are thus needed for P2P-based 3D streaming.

Here, we describe the P2P approach to 3D streaming with a conceptual model and examine some recent designs.<sup>5-7</sup> Using prototyping (see Figure 1) and simulations of our proposed framework, Flowing Level-of-Details (FLoD),<sup>7</sup> we show that P2P holds great promise for providing scalable and affordable content delivery for future 3D virtual worlds.

### 3D Streaming Requirements

To understand how 3D streaming could work on P2P networks, we must first identify its requirements and challenges from both the client and server perspectives. Four main types of 3D streaming exist today: object-, scene-, visualization-, and image-based.<sup>7</sup> In the context of virtual worlds, our main focus would be on scene streaming, whose goal is to provide each user a navigation experience within a scene by progressively delivering the 3D objects within the user's *area of interest* (AOI, or the area currently visible to the user, often denoted as a circle around him or her<sup>2</sup>). We can assume

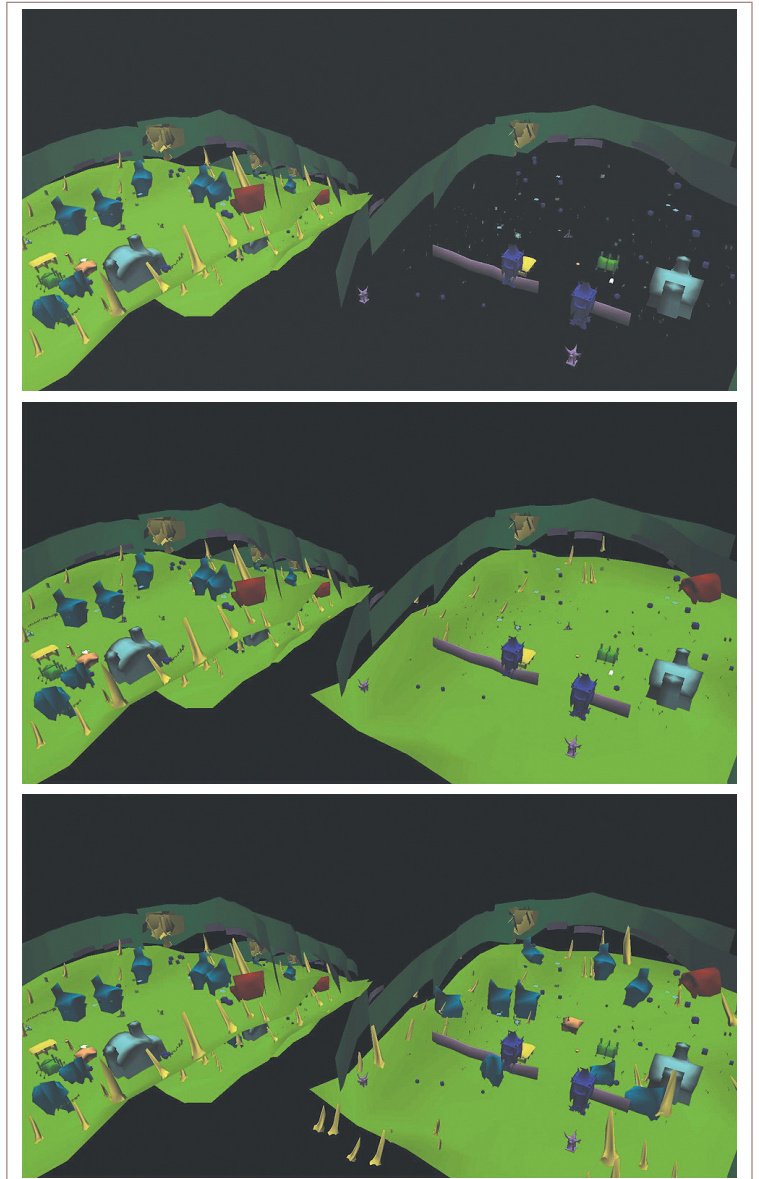


Figure 1. Peer-to-peer (P2P) 3D streaming prototype system. The different frames show the progression of streaming as time passes.

that each object consists of 3D models (such as meshes) and other associated data (for instance, textures, height maps, light maps, and animations), plus a certain position and orientation described in some form. To facilitate delivery, the content provider first fragments each object into a base piece and many refinement pieces in an application-specific manner at the server, using methods such as progressive meshes<sup>8</sup> or geometry images<sup>9</sup> for models, or progressive GIF, JPEG, or PNG formats for textures. Once the client obtains a set of base pieces, it can perform an initial scene rendering to allow timely navigation. Additional time in the scene will

## Related Work in 3D Scene Streaming

Researchers have proposed various techniques for transmitting a given 3D scene over a network to allow user interactions without preinstallation. Dieter Schmalstieg and Michael Gervautz were the first to introduce scene streaming in which a server determines and transmits visible objects at different levels-of-detail (LODs) to clients.<sup>1</sup> Eyal Teler and Dani Lischinski used prerendered, image-based impostors as the lowest LOD to enable faster initial visualizations.<sup>2</sup> Cyberwalk adopts progressive meshes to avoid the data redundancy from multiple LODs and focuses on caching and prefetching to enhance visual perception.<sup>3</sup> Social massively multiplayer online games (MMOGs) such as ActiveWorlds, There.com, and Second Life,<sup>4</sup> as well as the 3D instant messenger IMVU, use scene streaming to support dynamic content, but little is publicly known of their mechanisms. Our work complements these works with distributed deliveries.

Ketan Mayer-Patel and David Gotz present the concept of nonlinear media streaming,<sup>5</sup> in which interactive content (for example, images for a virtual museum) is divided and sent through multicast channels clients have subscribed to. The system supports a large number of receivers by send-

ing the content via application-layer multicast (ALM). However, ensuring proper content partitioning (so that clients receive only relevant content) and bounded latency (important for interactive applications) are nontrivial issues. Under this approach, clients might receive excessive content beyond current interests and experience variable latencies due to ALM channels' limitations.

### References

1. D. Schmalstieg and M. Gervautz, "Demand-Driven Geometry Transmission for Distributed Virtual Environments," *Computer Graphics Forum*, vol. 15, no. 3, 1996, pp. 421–432.
2. E. Teler and D. Lischinski, "Streaming of Complex 3D Scenes for Remote Walkthroughs," *Computer Graphics Forum*, vol. 20, no. 3, 2001, pp. 17–15.
3. J. Chim et al., "CyberWalk: A Web-Based Distributed Virtual Walkthrough Environment," *IEEE Trans. Multimedia*, vol. 5, no. 4, 2003, pp. 503–515.
4. P. Rosedale and C. Ondrejka, "Enabling Player-Created Online Worlds with Grid Computing and Streaming," *Gamasutra Resource Guide*, 2003; [www.gamasutra.com/resource\\_guide/20030916/rosedale\\_01.shtml](http://www.gamasutra.com/resource_guide/20030916/rosedale_01.shtml).
5. K. Mayer-Patel and D. Gotz, "Scalable, Adaptive Streaming for Nonlinear Media," *IEEE Multimedia*, vol. 14, no. 3, 2007, pp. 68–83.

let clients download and render models in more detail, given certain quality-of-service (QoS) requirements.<sup>10</sup> The two main requirements for a 3D streaming system are thus *streaming quality* and *scalability*.

### Streaming Quality

From the user's perspective, the main concern for 3D streaming is its visual quality. However, because visual quality can be subjective, a more definable concept is the streaming quality in terms of how much data a client obtains and how quickly. For the former, we could use the ratio between the data already downloaded and that required to render the current view – that is, a *fill ratio*. A ratio of 100 percent indicates the best visual quality (that is, the same as preinstalled content). For the latter, we can use base latency and completion latency to indicate the time taken to obtain an object's base piece or full data. Base latency indicates the delay in displaying a basic view of the scene, whereas completion latency indicates the time needed to fully inspect objects. The goal is thus to maximize fill ratios while minimizing latencies.

### Scalability

From the server's perspective, the main goal is to maximize the number of concurrent users by

distributing transmission and processing loads to clients as much as possible. For transmissions, it's preferable for clients, rather than the server, to deliver most content. For processing, the server should minimize its role in calculating transmission strategies. Ideally, if the server can fully delegate these calculations (for example, distributed determination of visibility and delivery prioritization) to clients, then it can simply answer data requests.

### Challenges in P2P 3D Streaming

Although it's relatively straightforward for a server to determine and deliver content to clients, switching to a P2P model introduces new challenges. In C/S 3D streaming, the server first performs *object discovery* for each client because it possesses complete knowledge of all objects and can determine each client's viewable objects. Given that only one data provider exists, *source discovery* isn't an issue. We can assume full data availability at the server, so the clients also don't need to perform any *state exchange* to learn of content availability. Finally, the content is transferred from the server to clients in a unidirectional manner. In P2P 3D streaming, however, we must re-examine these tasks as follows, while considering performance and scalability.

### Object Discovery

To know which objects to download, the user client must first discover the objects within its AOI. Preferably, the client should conduct visibility determination without the server being involved or having any global knowledge of the scene. However, because only the server initially possesses complete scene knowledge, we must partition and distribute the scene descriptions (that is, object metadata such as placement or orientation) so that the client can conduct a distributed discovery (via hierarchical trees<sup>5</sup> or grids,<sup>7,11</sup> for example).

### Source Discovery

To obtain content from other clients instead of the server, each client must know some other peers who might hold the content of interest. These partner peers likely are within each other's AOIs given that overlapped visibility indicates shared interests. However, other peers who have been in the same area previously might also retain content in their caches. So, how to maintain and discover potential content sources, either centrally<sup>6</sup> or in a distributed way,<sup>7</sup> is another challenge.

### State Exchange

Once the client finds a few peers, it still needs to know which content pieces are available at which peers, and what network conditions exist for each peer, so that it can fulfill content requests. A naïve approach is to query (that is, *pull*) each known peer,<sup>7</sup> but the query-response time might not meet 3D streaming's real-time requirements. A *push* approach, in which peers proactively notify each other about content availability, is thus also possible.<sup>12</sup>

### Content Exchange

To optimize the visual (or streaming) quality for a given bandwidth budget, a client can leverage its knowledge of peers to schedule content requests based on visibility, content availability, and network conditions. Interestingly, 3D streaming can be view-dependent,<sup>8</sup> with data pieces applied arbitrarily to reconstruct objects. As long as the download sequence satisfies the piece dependencies, only a roughly sequential transfer is needed (as opposed to the strictly sequential transfer required for video streaming). The clients can also exploit concurrent downloads to accelerate the retrieval for pieces

that don't involve dependencies. Depending on the results of initial requests, additional peers or requests might then be needed.

### A Conceptual Model

Given the requirements and challenges we've discussed, we categorized the main tasks for P2P-based 3D scene streaming as follows (see Figure 2):

- *Partitioning* divides the entire scene into smaller units so that the client doesn't require global knowledge of all object placements to determine visibility.<sup>5</sup> Scene partitioning is essential for decentralizing visibility calculations.
- *Fragmentation* divides 3D objects into pieces so that a client can progressively receive and reconstruct the 3D objects. Progressive meshes<sup>8</sup> and geometry images<sup>9</sup> are some example techniques.
- *Prefetching* predicts data usage ahead of time and generates data requests so that transmission latency is hidden from users. The client can also try to predict user movements or behaviors for this task.<sup>10,11</sup>
- *Prioritization* occurs when the client performs visibility determination to generate the request order for object pieces. The goal is to maximize visual quality by considering factors such as distance, line of sight, or visual importance.<sup>2,11</sup>
- *Selection* determines the proper peers to connect with and pieces to obtain based on peer capacity, content availability, and network conditions to efficiently fulfill prefetching and prioritization needs.

We can see from this model that the main additional tasks in P2P 3D streaming are partitioning the scene (for distributed visibility determination) and selecting peers and pieces (for P2P content exchange). Other tasks more or less are also required in C/S 3D streaming.

### FLoD Design

Based on the model just mentioned, we developed the FLoD framework<sup>7</sup> to realize P2P 3D streaming. Given that content and users (or nodes) tend to cluster at hotspots,<sup>13</sup> a user often has overlapped visibility with its AOI neighbors (that is, other users within that user's AOI). By requesting data from the neighbors first, the



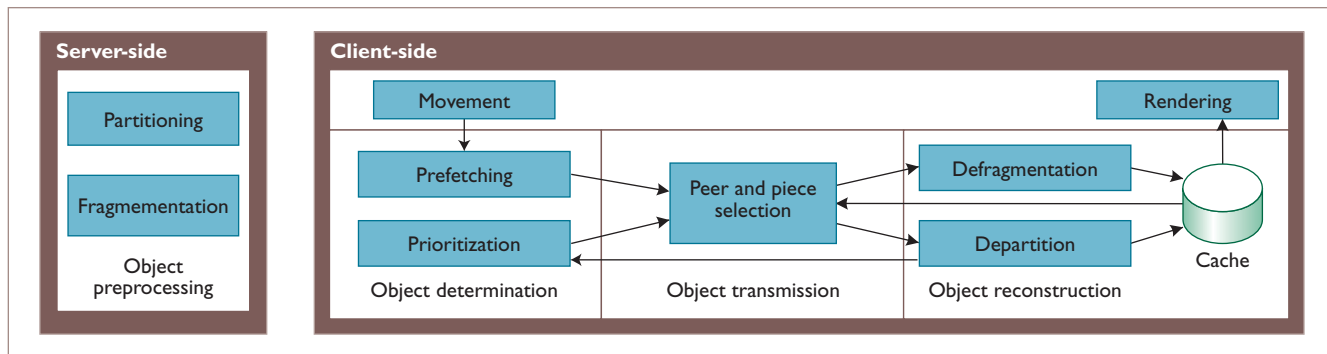


Figure 2. A conceptual model for peer-to-peer (P2P)-based 3D scene streaming. Obtaining movements and performing rendering are the only tasks required when content is locally available. Object preprocessing, determination, transmission, and reconstruction are additional stages for networked 3D streaming. Client-server 3D streaming requires only fragmentation, prefetching, and prioritization. Partitioning and selection are the new tasks required for P2P-based 3D streaming.

server needn't answer content requests repetitively. Note that neighbors here are based on proximity inside the virtual world and not on the physical network. Finding AOI neighbors is, in fact, the discovery of the proper interest groups for content exchange and must occur efficiently. For this purpose, we utilize a Voronoi-based Overlay Network (VON)<sup>14</sup> that supports neighbor discovery for P2P-VEs. VON lets a node learn its AOI neighbors' IDs, virtual coordinates, and IP addresses (akin to performing a spatial query for objects within the AOI). The basic idea is that even though a node might not know other nodes beyond its AOI, its neighbors near the AOI border (called *boundary neighbors*) have such knowledge and can send notification about incoming neighbors. We can thus conduct neighbor discovery in a fully distributed way without relying on any servers.

To allow client-side visibility determination, we partition the VE into square grids, each with a small scene description for the objects within. Each client can then determine the visible objects by retrieving scene descriptions for the cells that its AOI covers. When entering a new area, a client first prepares some scene requests to obtain scene descriptions from its AOI neighbors or the server. The client then performs object discovery by decoding the scene descriptions and sends out prioritized piece requests based on the client's visibility preferences. As data pieces arrive from either other peers or the server (which acts as the final data source for unfilled peer requests), the client can render a view progressively. Before the actual content exchange, the client first queries its neighbors on content availability. Among the neighbors that answer

positively, some are chosen randomly to request the actual content. The client repeats this process as it moves in the environment.

To demonstrate how FLoD works in real scenarios, we implemented a prototype (see Figure 1) that performs all the major 3D streaming tasks except prefetching. We experimented with the prototype by setting up a Linux server to load the initial scene and respond to client requests as users log in to navigate the scenes. The experiment shows that the server bandwidth usage is about half that of a pure C/S approach because clients can be self-sufficient in content serving.<sup>7</sup>

To investigate large-scale behaviors, we then performed simulations with bandwidth limits (a 1-Mbps download/256-Kbps upload limit for clients, and a 10-Mbps symmetric limit for the server). We placed objects randomly on a 2D map, with sizes based on our prototype (15 Kbytes per object, with a 3-Kbyte base piece, 1.2-Kbyte refinement pieces, and 300 to 500 bytes per scene description). The nodes move with constant speeds using random waypoints<sup>11</sup> for 3,000 time steps and request scene descriptions or data pieces as needed.

Scalable systems must keep resource usages bounded at all relevant system nodes. Figure 3a shows the upload bandwidth for both a C/S server and a FLoD server. Because the server upload limit is 10 Mbps, the C/S server exhausts its bandwidth at 1,250 Kbyte/sec when serving 200 nodes. On the other hand, a FLoD server's upload stays relatively constant under 50 Kbytes/sec. Figure 3b explains this reduction, showing that the upload and download bandwidths of FLoD clients converge, indicating

that as the system scales (that is, as the number of AOI neighbors increases), FLoD clients become self-sufficient in content serving. However, some bandwidth overhead exists for using the P2P overlay, because the overlay needs to exchange user positions and notify peers of new neighbors.<sup>7</sup> Although this overhead indicates a certain scalability limit as AOI neighbor density increases, the entire system can still accommodate users scalably (to possibly millions of concurrent users).

### Comparisons and Open Questions

FLoD addresses object discovery by using grid-based scene descriptions, and queries AOI neighbors from a P2P-VE overlay for source discovery. It uses a query-response approach for state exchange and random selection from peers for content exchange. Although this basic design is simple, the data sources are limited and the query for states might be slow. In a subsequent work,<sup>12</sup> we let clients keep historic AOI neighbors as extra sources and proactively push content availability to all connected neighbors to reduce the query-response delay. Clients also send requests to closer AOI neighbors first to avoid concentrating requests. Simulation results show that both fill ratio and base latency have improved.

Other researchers have recently proposed two alternative designs for P2P 3D streaming. Table 1 shows a taxonomy based on the main challenges we mentioned and compares these designs with FLoD, while outlining the potential solution space.

#### LODDT

The level-of-detail description tree (LODDT)<sup>5</sup> is a tree structure that stores urban cityscapes hierarchically. Clients can progressively perform visibility determination given a partial tree covered by the user's AOI. LODDT also evaluates a few peer-selection strategies based on object proximity and estimated content availability. Object discovery is thus based on a distributed tree, whereas source discovery is performed with a P2P-VE overlay similar to FLoD. However, given that only a selected set of connectivity peers provides the AOI neighbors, LODDT is more of a super peer than a fully distributed design. To learn about client states, peers also exchange queries and responses. Content availability isn't exchanged but is rather inferred

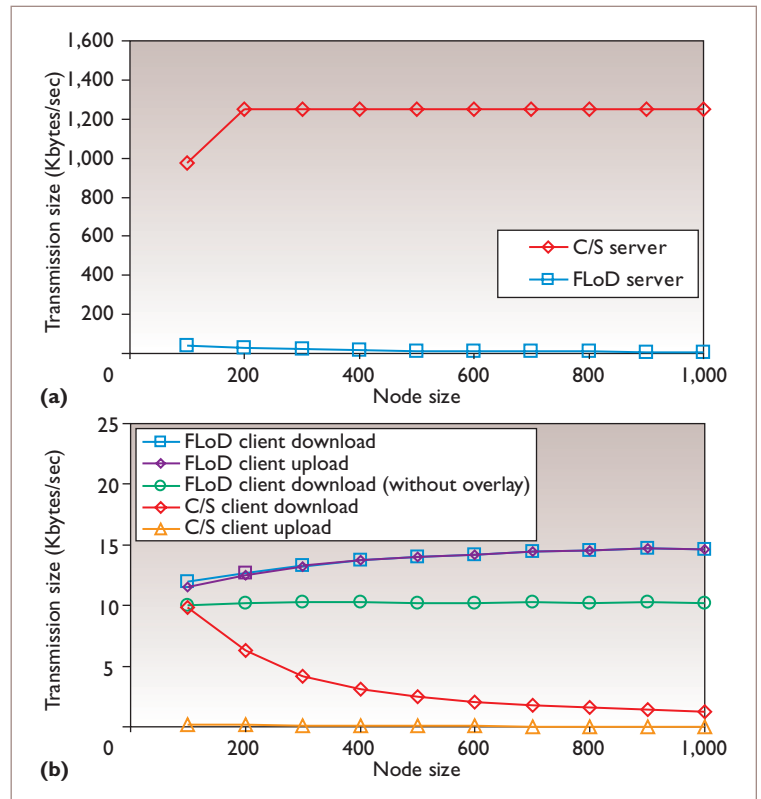


Figure 3. Bandwidth usage comparison between peer-to-peer (P2P) and client-server (C/S) 3D streaming. (a) Server bandwidth usage shows that a P2P server uses much less bandwidth than a C/S server after a saturation point of 200 nodes. (b) Client bandwidth usage shows that by providing content through client upload, a P2P client has matching upload/download sizes, which alleviates the server from content transmission. Some overhead for using the overlay exists and grows with user density in the region.

from the relative positions between neighbors. Based on response time and estimated content availability, a client then randomly makes requests from potential sources.

#### HyperVerse

HyperVerse uses a group of public servers to construct a static, structured overlay that maintains user positions for a VE.<sup>6</sup> The clients learn of other peers from the servers and exchange content by forming a loosely structured overlay. Thus, the server performs object and source discovery centrally and notifies clients of relevant peers and objects. No explicit state exchange policy exists, and clients request content from random neighbors.

#### Object Discovery Comparison

FLoD differs from LODDT mainly in the scene-partitioning method (for example, FLoD uses

Table 1. Taxonomy of P2P 3D streaming approaches.

Stage	FLoD		Level-of-detail description tree (LODDT)	HyperVerse
	Basic	Enhanced		
Object discovery	Grid-based scene descriptions	Grid-based scene descriptions	Hierarchical scene descriptions	Server-provided list
Source discovery	AOI (area of interest) neighbors (from peers)	Extended AOI neighbors (from peers)	$n$ nearest neighbors (from super peers)	AOI neighbors (from server)
State exchange	Query-response (pull)	Incremental update (push)	Query-response (pull)	N/A
Content exchange	Random selection	Multilevel AOI selection	Round-trip time (RTT) and estimated peer loading	Random selection

grids whereas LODDT utilizes trees); HyperVerse uses the server to maintain the object list. A central list is arguably more flexible and secure because distributed scene descriptions aren't straightforward to update, and malicious clients could manipulate the object list. On the other hand, a central list faces scalability limitations if the server receives too many requests. Grid partitioning is simple, and only a few cells are needed for ground-level navigation. However, for scenes viewable from different altitudes (such as city models or virtual globes), grids become inefficient because too many cells might be involved. Tree structures, however, require a top-down construction, so clients must first retrieve many nodes from the root down before they can determine ground-level objects.

#### Source Discovery Comparison

The current designs use mostly AOI neighbors as sources, maintained either centrally or among peers. Using a P2P-VE overlay for neighbor discovery can drastically reduce server loads.<sup>14</sup> However, the overlay incurs some overhead that grows with AOI neighbor density. A super-peer-based tracker for sources might be a balance between the two extremes and could keep track of non-AOI neighbors with relevant content. However, we must consider the fault-tolerance of the super-peers, so that the neighbor tracking tasks would not become unavailable when the super-peers fail.

#### State Exchange Comparison

The current designs exchange few states (such as content availability and network condition), and the two main approaches are pull (query-response) and push (proactive update). Current evaluation indicates that the push approach is faster than the pull approach.<sup>12</sup> However,

whether alternative or hybrid approaches exist requires further investigation.

#### Content Exchange Comparison

Both the basic FLoD design and HyperVerse use random selection. With enhanced FLoD, clients send requests to closer neighbors first, whereas LODDT bases requests on estimated capacities. However, researchers have yet to make detailed comparisons among these approaches. The clients can also use additional criteria to form the requests, such as latency or piece dependency. Current methods are mostly pull-based (that is, a client sends requests to the source providers), but push-based approaches are also possible (that is, sources proactively send out content, similar to how content delivery networks, or CDNs, push Web content to different geographic servers).

Besides these network-specific issues, other 3D streaming requirements are also worth exploring – for example, commercial applications likely will require dynamic updates and content authentication.<sup>15</sup> Prefetching and caching are also important aspects we have yet to investigate in-depth.<sup>10,11</sup>

A recent study on Second Life traffic has shown that 60 to 88 percent of server bandwidth usage is for textures, and a busy region might deliver close to 100 Gbytes of textures a day.<sup>13</sup> As our experiments and simulations show, FLoD can relieve the server of its heavy loads. Virtual globe applications such as Google Earth might also benefit from P2P 3D streaming as they move toward 3D content with possible multiuser interactions.

Real-time 3D content has yet to find its way to most Internet users in spite of years of effort. Although challenges remain in areas such as

protocol standards and ease of content creation, content streaming can effectively address the delivery problem. 3D streaming on P2P networks is thus a topic of interest to both graphics and networking professionals. By identifying the basic issues, we hope to highlight this promising direction for making 3D content more accessible. An open source implementation of FLoD is available at <http://ascend.sourceforge.net>. □

### Acknowledgments

This work was supported by National Science Council Taiwan, ROC., under grants 95-2221-E-008-048-MY3 and 95-2221-E-002-273-MY2. We thank Shing-Tsaan Huang for his support and Shao-Chen Chang and Ting-Hao Huang for invaluable discussions. We're grateful to the National Center for High-Performance Computing (NCHC), Taiwan, for its computing facilities, Guan-Ming Liao for the game scene, and the constructive feedback from the anonymous reviewers. We also thank Nein-Hsien Lin, Wei-Lun Sung, Guan-Yu Huang, and Chien-Hao Chien.

### References

1. P. Rosedale and C. Ondrejka, "Enabling Player-Created Online Worlds with Grid Computing and Streaming," *Gamasutra Resource Guide*, 2003; [www.gamasutra.com/resource\\_guide/20030916/rosedale\\_01.shtml](http://www.gamasutra.com/resource_guide/20030916/rosedale_01.shtml).
2. D. Schmalstieg and M. Gervautz, "Demand-Driven Geometry Transmission for Distributed Virtual Environments," *Computer Graphics Forum*, vol. 15, no. 3, 1996, pp. 421-432.
3. E. Teler and D. Lischinski, "Streaming of Complex 3D Scenes for Remote Walkthroughs," *Computer Graphics Forum*, vol. 20, no. 3, 2001, pp. 15-17.
4. K. Mayer-Patel and D. Gotz, "Scalable, Adaptive Streaming for Nonlinear Media," *IEEE Multimedia*, vol. 14, no. 3, 2007, pp. 68-83.
5. J. Royan et al., "Network-Based Visualization of 3D Landscapes and City Models," *IEEE Computer Graphics and Applications*, vol. 27, no. 6, 2007, pp. 70-79.
6. J. Botev et al., "The HyperVerse – Concepts for a Federated and Torrent Based '3D Web,'" *Int'l J. Advanced Media and Communication*, vol. 2, no. 4, 2008, pp. 331-350.
7. S.-Y. Hu et al., "FLoD: A Framework for Peer-to-Peer 3D Streaming," *Proc. 27th Conf. Computer Communication (INFOCOM 08)*, IEEE CS Press, 2008, pp. 1373-1381.
8. H. Hoppe, "View-Dependent Refinement of Progressive Meshes," *Proc. 24th Ann. Conf. Computer Graphics and Interactive Techniques (SIGGRAPH 97)*, ACM Press, 1997, pp. 189-198.
9. N.-S. Lin, T.-H. Huang, and B.-Y. Chen, "3D Model Streaming Based on JPEG 2000," *IEEE Trans. Consumer Electronics*, vol. 53, no. 1, 2007, pp. 182-190.
10. G.V. Popescu and C.F. Codella, "An Architecture for QoS Data Replication in Network Virtual Environments," *Proc. IEEE Virtual Reality Conf. (VR 02)*, IEEE Press, 2002, pp. 41-48.
11. J. Chim et al., "CyberWalk: A Web-Based Distributed Virtual Walkthrough Environment," *IEEE Trans. Multimedia*, vol. 5, no. 4, 2003, pp. 503-515.
12. W.-L. Sung, S.-Y. Hu, and J.-R. Jiang, "Selection Strategies for Peer-to-Peer 3D Streaming," *Proc. 18th Int'l Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 08)*, ACM Press, 2008, pp. 15-20.
13. H. Liang, M. Motani, and W.T. Ooi, "Textures in Second Life: Measurement and Analysis," *Proc. 14th IEEE Int'l Conf. Parallel and Distributed Systems, Workshop P2P-NVE (ICPADS 08)*, IEEE Press, 2008, pp. 823-828.
14. S.-Y. Hu, J.-F. Chen, and T.-H. Chen, "VON: A Scalable Peer-to-Peer Network for Virtual Environments," *IEEE Network*, vol. 20, no. 4, 2006, pp. 22-31.
15. M.-C. Chan, S.-Y. Hu, and J.-R. Jiang, "Secure Peer-to-Peer 3D Streaming," *Multimedia Tools and Applications*, vol. 45, nos. 1-3, Oct. 2009, pp. 369-384.

---

**Shun-Yun Hu** is a research assistant at the Institute of Information Science, Academia Sinica, Taiwan. His research interests include networked virtual environments and peer-to-peer systems. Hu has a PhD in computer science from National Central University, Taiwan. He's a member of the IEEE and the ACM. Contact him at [syhu@ieee.org](mailto:syhu@ieee.org).


---

**Jehn-Ruey Jiang** is an associate professor in the Department of Computer Science and Information Engineering, National Central University, Taiwan. His research interests include peer-to-peer computing and pervasive computing. He's a member of the IEEE and the ACM. Contact him at [jrjiang@csie.ncu.edu.tw](mailto:jrjiang@csie.ncu.edu.tw).

---

**Bing-Yu Chen** is an associate professor in the Department of Information Management, the Department of Computer Science and Information Engineering, and the Graduate Institute of Networking and Multimedia at the National Taiwan University, and a visiting associate professor in the Department of Computer Science at the University of Tokyo. His research interests are mainly in computer graphics, human-computer interaction, and image and video processing. Chen has a PhD in information science from the University of Tokyo. He's a member of the IEEE, the ACM, ACM SIGGRAPH, Eurographics, IEICE, and IICM. Contact him at [robin@ntu.edu.tw](mailto:robin@ntu.edu.tw).

---

 Selected CS articles and columns are also available for free at <http://ComputingNow.computer.org>.