To appear in IEEE Internet Computing

Peer-to-Peer 3D Streaming

Shun-Yun Hu, Jehn-Ruey Jiang, and Bing-Yu Chen*

National Central University
National Taiwan University*

Abstract Virtual worlds have become very popular in recent years, with trends towards 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. We define both the problem and solution spaces for P2P 3D streaming, by outlining its requirements, challenges, and categorizing existing proposals. Open questions are also identified to facilitate the design of future systems.

Keywords: 3D streaming, peer-to-peer, virtual environments

Multiuser 3D virtual environments (VEs) such as Massively Multiplayer Online Games (MMOGs) have become very popular in recent years. *World of Warcraft*, a role-playing MMOG where players participate in epic battles and adventures between two opposing camps, has over 11 million paying subscribers by 2008[1]. *Second Life* [1], a social MMOG entirely built by its "residents", also boosts over 1.4 million active accounts and nearly 9 million U.S. dollar worth of virtual item transactions each month[2]. 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 now building virtual worlds (e.g., *Second Life* hosts 34 terabytes of user-generated content in 2007[3]). The trends towards *larger worlds* and *larger numbers of worlds* are beginning to show the inadequacy of current VE installation methods, which require users to pre-install the full content via DVDs or a prior download. Easier access to the various and massive amount of VE content thus demands *3D streaming* for content distribution [2][3].

---

[1] http://www.blizzard.com/us/press/081028.html
[2] http://secondlife.com/whatis/economy_stats.php
[3] http://www.informationweek.com/news/showArticle.jhtml?articleID=197800179

3D streaming is a technique that delivers 3D content over networks in real-time to allow users to navigate a VE without a complete content installation. Similar to audio or video *media streaming*, 3D streaming requires the content be fragmented into pieces before it can be transmitted, reconstructed, and displayed. However, unlike the linearly streamed audio or video, 3D streaming is highly interactive and non-linear in nature, where the streaming sequence is based on the visibility or interest area of each user, making it individually unique among the users.

Current 3D streaming schemes adopt the *client-server* model for content delivery. However, such architecture is difficult to scale as 3D streaming is both data- and processing-intensive. Prohibitively vast amount of server-side bandwidth and CPU power are required for a massive audience. On the other hand, highly scalable yet affordable computing and content sharing systems have been built successfully with *peer-to-peer* (P2P) networks. As users in a 3D scene could own similar content due to overlapped visibility, they may obtain content from each other in a P2P manner. However, while P2P media streaming has progressed significantly in recent years, it may 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 [4]. New insights and novel designs are thus needed for P2P-based 3D streaming.

In this article, we describe the P2P approach to 3D streaming with a conceptual model and some recent designs [5] [6] [7]. Through prototyping (see Fig. 1) and simulations of our proposed framework, FLoD [7], we show that P2P holds great promises to provide scalable and affordable content delivery for future 3D virtual worlds.

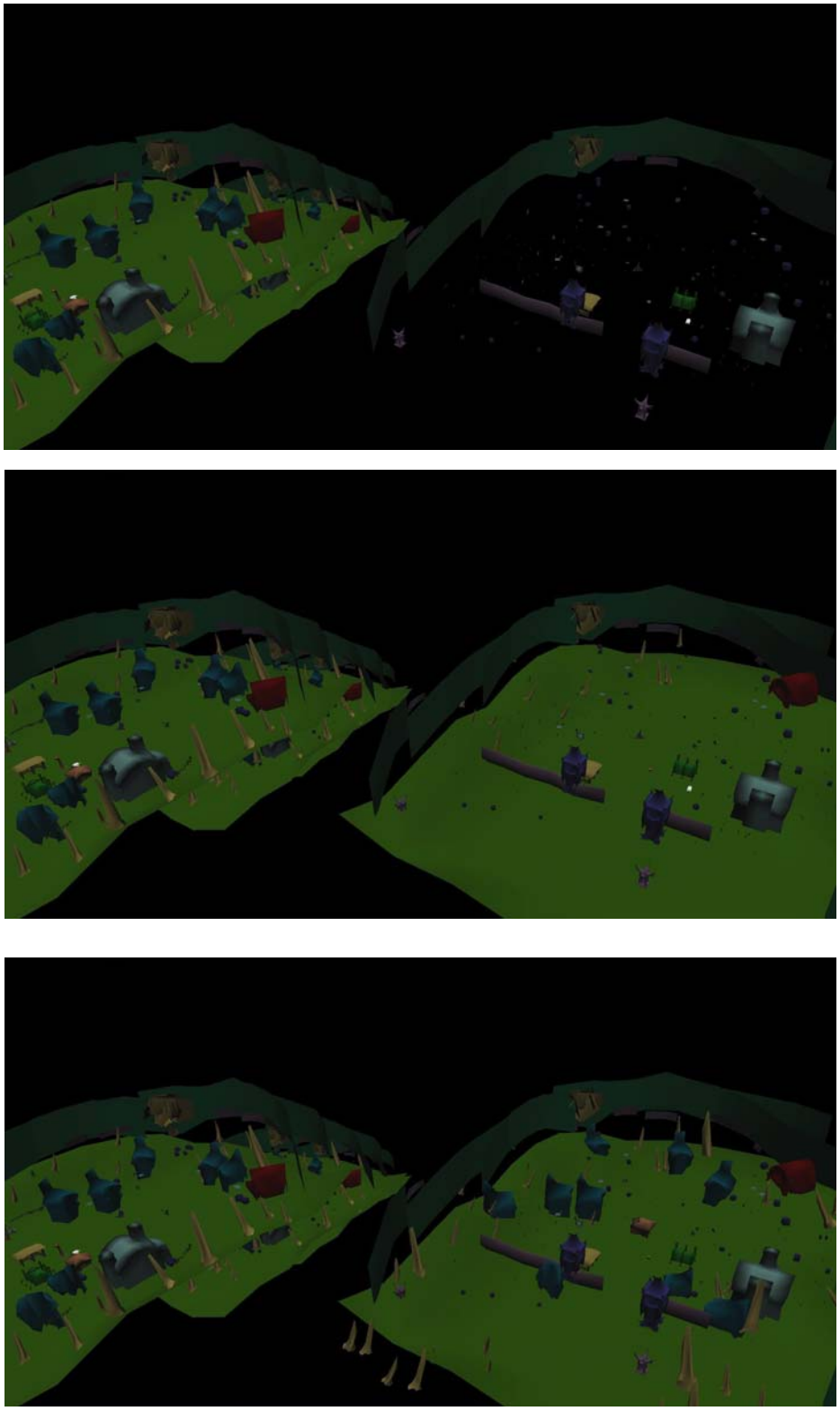


Fig. 1. P2P 3D streaming prototype system. The different frames show the progression of streaming as time passes.

3D Streaming Requirements

To understand how 3D streaming could work on P2P networks, we must first identify its requirements and challenges from the perspectives of both clients and servers. There are four main types of 3D streaming today: *object streaming, scene streaming, visualization streaming*, and *image-based streaming* [7]. 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) (i.e., the area currently visible to the user, often denoted as a circle around the user [2]). We may assume that each object consists of 3D models (e.g., meshes) and other associated data (e.g., textures, height maps, light maps, and animations), plus a certain position and orientation described in some form of *scene description*. To facilitate delivery, each object is first fragmented into a *base piece* and many *refinement pieces* in application-specific manners, using methods such as *progressive meshes* [8] or *geometry images* [9] for models, or progressive GIF, JPEG, or PNG for textures. Once a set of base pieces is obtained, an initial rendering of the scene can be performed to allow timely navigation. Additional time in the scene will allow users to download and render models in more details, given certain Quality of Service (QoS) requirements [10]. The two main requirements for a 3D streaming system thus are:

- **Streaming quality**

  From the user's perspective, the main concern for 3D streaming is its *visual quality*. However, as visual quality can be subjective, a more definable concept is the *streaming quality* in terms of "how much" and "how fast" a client obtains data. For the former, we could use the ratio between the data already downloaded and that required to render the current view (i.e., a *fill ratio*). A ratio of 100% indicates the best visual quality (i.e., the same as pre-installed content). For the latter, we may use *base latency* and *completion latency* to indicate the time to obtain the base piece or the full data of an object. Base latency indicates the delay to display a basic view of the scene, while completion latency indicates the time needed to fully inspect objects. The goal thus is to maximize fill ratios while minimizing the 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 is preferable if most content is delivered *by clients* and not by the server. For processing, the server should minimize its

role in calculating the transmission strategies. Ideally, if these calculations are fully delegated to clients (e.g., distributed determination of visibility and delivery prioritization), then the server can simply answer data requests.

Challenges in P2P 3D streaming

While it is relatively straightforward for a server to determine and deliver content to clients, new issues are introduced when we switch to a P2P model. We can understand them from the perspective of a client, who needs to consider the following issues:

- **Object Discovery:**
  In order to know which objects are to be downloaded, the user client must first discover the objects within its AOI. Preferably, visibility determination should be done without the server's involvement or any global knowledge of the scene. However, as only the server initially possesses complete knowledge of the scene, we need to partition and distribute the *scene descriptions* (i.e., object meta-data such as placements or orientations), so that a distributed discovery can be done (e.g., via hierarchical trees [5] or grids [7] [11]).

- **Source Discovery**:
  To obtain content from other user clients instead of the server, each client must know some other clients (i.e., *peers*) who may hold the content of interest. These partner peers likely are within each other's AOI as overlapped visibility indicates shared interests. However, other peers who have been in the same area previously may also retain content in cache. So how to maintain and discover potential content sources, either centrally [6] or in distributed manners [7], is another challenge.

- **State Exchange**:
  Once a few peers are found, we still need to know which content pieces are available at which peers, and what network conditions exist for each peers, so that content requests can be fulfilled. A naïve approach is to query each known peer [7]. However, the query-response time of such a *pull* approach may not meet the real-time requirements of 3D streaming. A *push* approach where peers proactively notify each other of content availability thus is also possible [12].

- **Content Exchange:**
  To optimize the visual (or streaming) quality for a given bandwidth budget, a client can then leverage its knowledge on peers to schedule content requests based on visibility, content availabilities and network conditions. Interestingly, as 3D streaming can be view-dependent [8], where data pieces may be applied arbitrarily to reconstruct objects. As long as the piece dependencies are satisfied, only a *roughly sequential* transfer is needed (as opposed to the *strictly sequential* video streaming). Concurrent downloads can also be exploited to accelerate the retrieval for pieces that do not involve dependencies. Depending on the results of initial requests, additional peers or requests may then be needed.

In client-server 3D streaming, object discovery is performed by the server, which possesses complete knowledge of all objects, and may determine for each client the viewable objects. Source discovery is not an issue as there is only one data provider. Full data availability is assumed for the server, and content delivery is unidirectional from the server to clients. In P2P 3D streaming, all of the above issues need to be addressed, with considerations to performance and scalability.

A Conceptual Model

Given the above requirements and challenges, we present a categorization of the main tasks for a P2P-based 3D scene streaming as follows (Fig. 2):

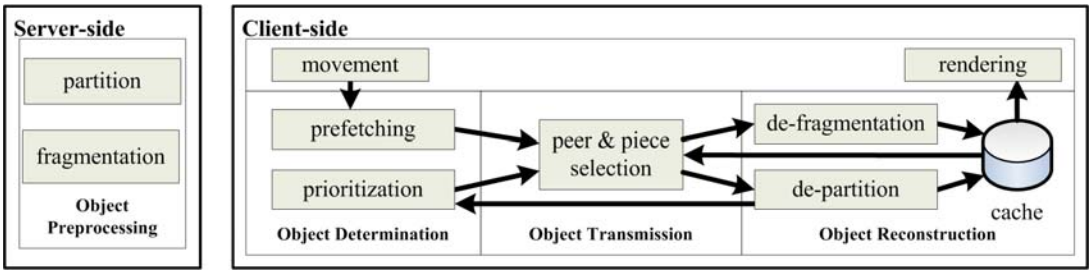


Fig. 2. A conceptual model for P2P-based 3D scene streaming, where obtaining movements and performing rendering are the only task when content is locally available. Object preprocessing, determination, transmission, and reconstruction are additional stages for networked 3D streaming. In client-server 3D streaming, only fragmentation, prefetching, and prioritization are needed. Partition and selection are the new tasks introduced for P2P-based 3D streaming.

- **Partition:** The task of dividing the entire scene into smaller units so that global knowledge of all object placements is not required for visibility determination [5]. Scene partition is essential if visibility calculations were to be decentralized.

- **Fragmentation:** The task of dividing 3D objects into pieces so that they may be transmitted via a network and reconstructed back progressively by a client. Progressive meshes [8] and geometry images [9] are some example techniques.

- **Prefetching:** The task of predicting data usage ahead of time and generating data requests so that transmission latency is masked from users. Predications of user movements or behaviors can be employed for this task [10] [11].

- **Prioritization:** The task of performing 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 [2] [11].

- **Selection:** The task of determining the proper peers to connect and pieces to obtain based on considerations of peer capacity, content availability and network conditions, in order to efficiently fulfill prefetching and prioritization needs.

The Design of FLoD

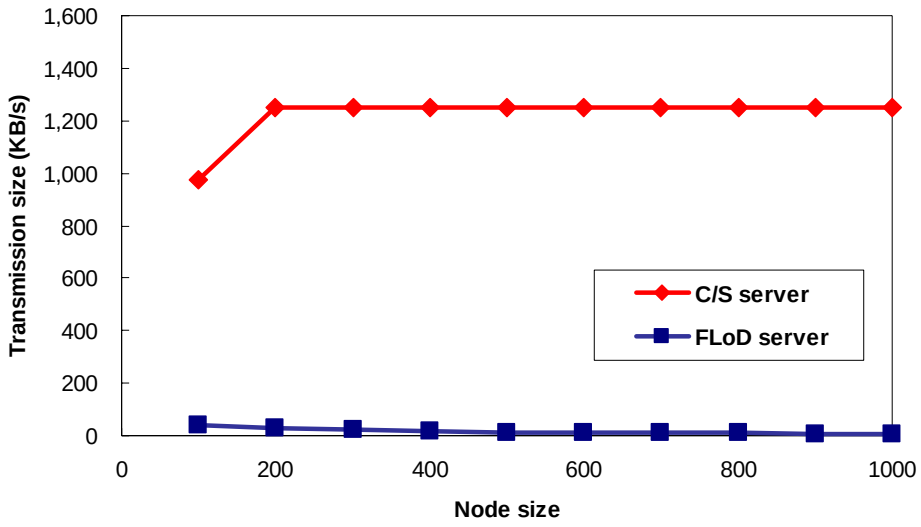
Based on the above model, we present the FLoD framework [7] to realize P2P 3D streaming. We observe that as content and users (or *nodes*) tend to cluster at *hotspots* [13], a user often has overlapped visibility with its *AOI neighbors* (i.e., other users who are within the given user's AOI). By requesting data from the neighbors first, the server can be relieved from serving content repetitively. Note that neighbors here are based on virtual proximity and not the physical network. Finding AOI neighbors is in fact the discovery of the proper *interest groups* for content exchange, and must be done efficiently. For this purpose, we utilize a *Voronoi-based Overlay Network* (VON) [14] that supports neighbor discovery for P2P virtual environments (P2P-VEs). VON allows a node to learn of its AOI neighbors' IDs, virtual coordinates, and IP addresses (i.e., akin to performing a *spatial query* for objects within the AOI). The basic idea is that even though a node may not know other nodes beyond its AOI, its neighbors near the AOI border (called the *boundary neighbors*) likely have such knowledge, and can notify about incoming neighbors. Neighbor discovery thus can be done in a fully distributed manner without relying on any servers.

To allow client-side visibility determination, the VE is partitioned into square grids, each of which has a small *scene description* for the objects within. Each client can then determine the visible objects by retrieving scene descriptions for cells covered by its AOI. When entering a new area, a client first prepares some *scene requests* to obtain scene descriptions from its AOI neighbors or the server. Prioritized *piece requests* based on preferences are then sent for the visible objects. A view is rendered progressively as data pieces arrive from either the peers or the server (which acts as the final data source for unfilled peer requests). A query is also made to neighbors on content availability before actual requests are sent to randomly selected neighbors that answer the query. The process is repeated as a client moves in the environment.

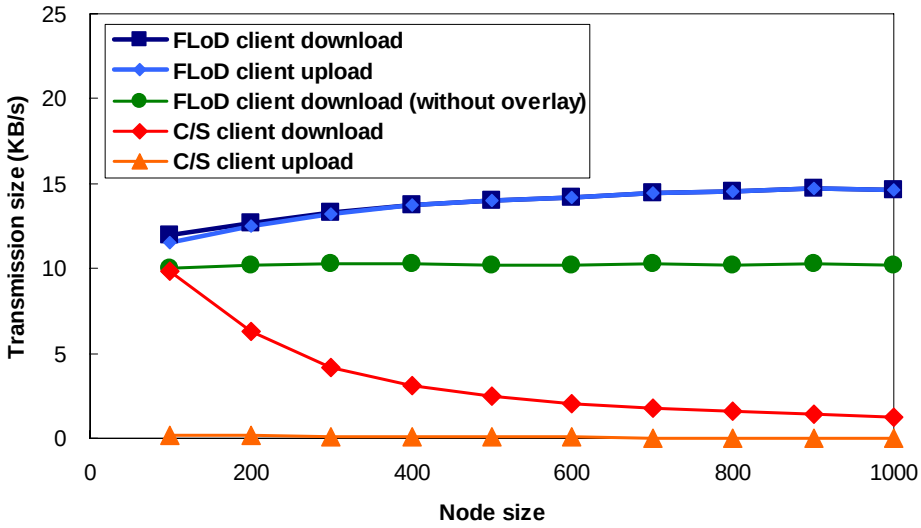
To demonstrate how FLoD works in real scenarios, we implement a prototype (Fig. 1) that performs all the major 3D streaming tasks except prefetching. We experiment with the prototype by setting up a Linux server to load the initial scene and respond to client requests, as users login to navigate and explore the scenes. The experiment shows that the server bandwidth usage is about half of a pure client-server approach, as clients can be self-sufficient in content serving [7].

To investigate large-scale behaviors, we then perform simulations with bandwidth limits (i.e., 1 Mbps download / 256 Kbps upload for clients, and 10 Mbps symmetric for the server). Objects are placed randomly on a 2D map, with sizes based on our prototype (i.e., 15 KB per object, with 3 KB base piece and 1.2 KB refinement pieces, and 300 to 500 bytes per scene description). The nodes move with constant speeds using random way-points [11] for 3000 time-steps, and request scene descriptions or data pieces as needed.

Scalable systems need to keep resource usages bounded at all relevant system nodes. Fig. 3(a) shows the upload bandwidth for both a client-server (C/S) server and a FLoD server. As the server upload limit is 10 Mbps, the C/S server exhausts its bandwidth at 1250 KB/s when serving 200 nodes. On the other hand, a FLoD server's upload stays relatively constant below 50 KB/s. This reduction is explained in Fig. 3(b), which shows that the upload and download bandwidth of FLoD clients converge, indicating that as the system scales (i.e., the number of AOI neighbors increases), FLoD clients become self-sufficient in content serving. However, some bandwidth overhead exists for using the P2P overlay, as the overlay needs to exchange user positions and notify peers of new neighbors [7]. This overhead thus grows as AOI neighbor density increases. While this overhead growth indicates a certain scalability limit, it also means that if the user density in each region is controlled, the entire system can accommodate users in a scalable manner.



(a)



(b)

Fig. 3. Bandwidth usage comparison between 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 that alleviates the server from content transmission. Some overhead for using the overlay exists and grows with user density in the region.

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. A query-response approach is used for state exchange, and random selection from peers for content exchange. Though this basic design is simple, the data sources are limited and the query for content availability may be slow. In a subsequent work [12], we let clients keep historic AOI neighbors as extra sources, and proactively push content availability to AOI neighbors to reduce the query-response delay. Piece requests are also sent to closer AOI neighbors first to avoid concentrating requests. Simulation results show that both fill ratio and base latency improve with the enhancements. Two recent designs have since been proposed for P2P 3D streaming, and represent alternative approaches in the solution space:

LODDT: *Level-of-detail description tree* (LODDT) [5] is a tree structure that stores urban cityscape hierarchically. Clients thus can progressively perform visibility determination given a partial tree covered by the user's AOI. A few peer selection strategies based on object proximity and estimated content availability are also evaluated. Object discovery thus is based on a distributed tree, while source discovery is performed with a P2P-VE overlay similar to FLoD. However, as 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 of the client states, queries and responses are also exchanged among the peers. Content availability is not exchanged, but inferred from the relative positions between neighbors. Based on response time and estimated content availability, requests are then made randomly to potential sources.

HyperVerse: *HyperVerse* [6] utilizes a group of public servers to construct a static, structured overlay that maintains the user positions for a VE. The clients learn of other peers from the servers, and exchange content by forming a loosely-structured overlay. Object discovery and source discovery thus are performed centrally by the server that notifies clients of relevant peers and objects. There is no explicit state exchange policy, while content is also requested from random neighbors.

We compare these designs with FLoD in a taxonomy based on the main challenges below (Fig. 4), while outlining the potential solution space for P2P 3D streaming:

Object Discovery: FLoD differs from LODDT mainly in the scene partitioning method (e.g., FLoD uses grids and LODDT utilizes trees), while HyperVerse uses the server to maintain the object list. A central list is arguably more flexible and secure, as distributed scene descriptions are not 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 small number of cells are needed for ground-level navigation. However, for scenes viewable from different altitudes (e.g., city models or virtual globes), grids become inefficient as too many cells can be involved. On the other hand, tree structures require a top-down construction; so many nodes from the root down have to be retrieved first, before ground-level objects can be determined.

Source Discovery: The current designs mostly use AOI neighbors as sources, maintained either centrally or among the peers. Using a P2P-VE overlay for neighbor discovery can drastically reduce server loads [14]. However, the overlay incurs some overhead that grows with AOI neighbor density. A super-peer-based tracker for sources may be a balance between the two extremes, and could keep track of non-AOI neighbors with relevant content. However, its failure needs to be well considered.

State Exchange: Few states are being exchanged among the current designs (e.g., content availability and network condition), and the two main approaches are *pull* (i.e., query-response) and *push* (i.e., proactive update). Current evaluation indicates that the push approach is faster than pull [12]. However, whether alternative or hybrid approaches exist warrants further investigations.

Content Exchange: Random selection is being used by both the basic FLoD design and HyperVerse. In the enhanced FLoD, requests are sent to closer neighbors first, while LODDT bases requests on estimated capacities. However, detailed comparisons have yet been made among these approaches. Additional criteria may also be used to form the requests, such latency or piece dependency. Current methods are mostly *pull*-based (i.e., requests are sent to the source providers), but *push*-based approaches is also a possibility (i.e., 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, additional requirements in 3D streaming are also worth exploring. For example, performing content update with distributed peers, or providing content authentication for commercial applications [15]. Content prefetching and caching is also an important aspect to 3D streaming [10] [11].

Approach / Stage	FLoD		LODDT	HyperVerse
	Basic	**Enhanced**		
Object Discovery	Grid-based scene descriptions		Hierarchical scene descriptions	Server-provided list
Source Discovery	AOI 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	Multi-level AOI selection	RTT and estimated peer loading	Random selection

Fig. 4. Taxonomy of P2P 3D streaming approaches. FLoD [7] relies on grid-partitioned scene descriptions and a fully-distributed P2P-VE overlay to discover objects and sources, while using both push and pull to exchange content. LODDT [5] utilizes a hierarchical tree to partition the scene, while prioritizing requests based on network conditions. HyperVerse [6] designates the discovery phase as a server task, and uses clients only for content exchange purposes.

Related Work

Schmalstieg and Gervautz [2] first introduce scene streaming where a server determines and transmits visible objects at different *level-of-details* (LODs) to clients. Teler and Lischinski use pre-rendered image-based *impostors* as the lowest LOD to allow faster initial visualizations [3]. *Cyberwalk* [11] adopts progressive meshes to avoid the data redundancy from multiple LODs, and focuses on caching and prefetching to enhance visual perceptions. Social MMOGs such as *ActiveWorlds*[4], *There.com*[5], and *Second Life* [1], as well as the 3D Instant messenger *IMVU*[6], utilize scene streaming to support dynamic content, but little of their mechanisms are publicly known. Our work complements the above works with distributed deliveries. Mayer-Patel and Gotz present the concept of *non-linear media streaming* [4] where interactive content (e.g., images for a virtual museum) is divided and sent through multicast channels subscribed by clients. The system supports a large number of receivers by sending the content via *application-layer multicast* (ALM). However, ensuring proper content partition (so that only relevant content is received) and bounded latency (important for interactive applications) are non-trivial issues. Under

[4] http://www.activeworlds.com/
[5] http://www.there.com/
[6] http://www.imvu.com/

their approach, the clients would receive excessive content beyond current interests and experience variable latencies due to the limitations of ALM channels.

Conclusion

FLoD represents one of the first steps towards P2P-based 3D streaming. Compared with P2P media streaming, P2P 3D streaming faces the challenges of non-linear content access and the latency-sensitive dynamic grouping of users with similar interests. Grouping in P2P media streaming is easier, as group membership is relatively static and well-defined. Compared with client-server 3D streaming, P2P 3D streaming must deal with the additional challenges of object discovery, source discovery, state exchange, and content exchange. A recent study on *Second Life* traffic has shown that 60% to 88% of server bandwidth usage is for textures, and a busy region may serve close to 100 GB of textures each day [13]. As shown by our experiments and simulations, FLoD indeed can relieve the server of its heavy loads. Virtual globe applications such as *Google Earth* may also benefit from P2P 3D streaming, as they move towards 3D content with possibly multi-user interactions[7].

Real-time 3D content has yet found a way to most Internet users in spite of years of efforts. While challenges remain in areas such as protocol standards and the ease of content creation, content streaming may effectively address the delivery problem. 3D streaming on P2P networks thus is 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 NSC, Taiwan, R.O.C. grant 95-2221-E-008-048-MY3 and 95-2221-E-002-273-MY2. We thank Prof. Shing-Tsaan Huang for his supports, and both Shao-Chen Chang and Ting-Hao Huang for invaluable discussions during FLoD's design. We are grateful of NCHC for the computing facilities, Guan-Ming Liao for the game scene, and the constructive feedbacks by the anonymous reviewers. We also would like to thank the following persons for their contributions on FLoD's evaluation: Nein-Hsien Lin, Wei-Lun Sung, Guan-Yu Huang, and Chien-Hao Chien.

[7] http://www.technologyreview.com/Infotech/18911

References

[1] Philip Rosedale and Cory Ondrejka, "Enabling Player-Created Online Worlds with Grid Computing and Streaming," *Gamasutra Resource Guide*, 2003.

[2] Dieter Schmalstieg and Michael Gervautz, "Demand-Driven Geometry Transmission for Distributed Virtual Environments," *Computer Graphics Forum*, vol. 15, no. 3, 1996, pp. 421—432.

[3] Eyal Teler and Dani Lischinski, "Streaming of complex 3D scenes for remote walkthroughs," *Computer Graphics Forum*, vol. 20, no. 3, 2001, pp. 17—15.

[4] Ketan Mayer-Patel and David Gotz, "Scalable, Adaptive Streaming for Nonlinear Media," *IEEE Multimedia*, vol. 14, no. 3, 2007, pp. 68—83.

[5] Jerome Royan et al., "Network-Based Visualization of 3D Landscapes and City Models," *IEEE CG&A*, vol. 27, no. 6, 2007, pp. 70-79.

[6] Jean Botev et al., "The HyperVerse - Concepts for a Federated and Torrent Based '3D Web'," *Intl. Journal of Adv. Media and Comm.*, Vol. 2, No.4, 2008.

[7] Shun-Yun Hu et al., "FLoD: A Framework for Peer-to-Peer 3D Streaming," in *Proc. INFOCOM*, 2008, pp. 1373-1381.

[8] Hugues Hoppe, "View-dependent refinement of progressive meshes," in *Proc. SIGGRAPH*, 1997, pp. 189—198.

[9] Nien-Shien Lin, Ting-Hao Huang, and Bing-Yu Chen, "3D Model Streaming based on JPEG 2000," *IEEE Trans. Consumer Electronics*, vol. 53, no. 1, 2007, p.182—190.

[10] George V. Popescu and Christopher F. Codella, "An Architecture for QoS Data Replication in Network Virtual Environments," in *Proc. IEEE Virtual Reality (IEEE VR)*, pp. 41-48, 2002.

[11] Jimmy Chim et al., "CyberWalk: A Web-Based Distributed Virtual Walkthrough Environment," *IEEE Trans. Multimedia*, vol. 5, no. 4, 2003, pp. 503—515.

[12] Wei-Lun Sung, Shun-Yun Hu, and Jehn-Ruey Jiang, "Selection Strategies for Peer-to-Peer 3D Streaming," in *Proc. NOSSDAV*, May 2008.

[13] Huiguang Liang, Mehul Motani, and Wei Tsang Ooi, "Textures in Second Life: Measurement and Analysis," in *Proc. IEEE ICPADS workshop P2P-NVE*, Dec. 2008.

[14] Shun-Yun Hu, Jui-Fa Chen and Tsu-Han Chen, "VON: A scalable peer-to-peer network for virtual environments," *IEEE Network*, vol. 20, no. 4, 2006, pp. 22—31.

[15] Mo-Che Chan, Shun-Yun Hu, and Jehn-Ruey Jiang, "Secure Peer-to-Peer 3D Streaming," to appear in *Multimedia Tools and Applications*, 2009.

Shun-Yun Hu is currently a Ph.D candidate at National Central University, Taiwan. He has been researching on how to create million-scale virtual world systems using peer-to-peer (P2P) techniques since 2003. His works have appeared in *IEEE Network* and *IEEE INFOCOM*, and are publicly available as the *SourceForge* projects VAST (http://vast.sourceforge.net) and ASCEND (http://ascend.sourceforge.net). To promote the understanding of scalable peer-to-peer virtual environments, he also co-organized the *Massively Multiuser Virtual Environments* (MMVE) workshop in 2008 and 2009 (http://peers-at-play.org/MMVE09/). He was inspired by Sierra adventure games at the age of 7 and has been fascinated by the possibilities of virtual worlds ever since. He is a student member of IEEE and ACM.

Jehn-Ruey Jiang received his PhD degree in Computer Science in 1995 from National Tsing-Hua University, Taiwan, R.O.C. He joined Chung-Yuan Christian University as an Associate Professor in 1995. He joined Hsuan-Chuang University in 1998 and became a full Professor in 2004. He is currently with the Department of Computer Science and Information Engineering, National Central University, Taiwan. He was a recipient of Best Paper Award of the 32nd International Conference on Parallel Processing, 2003, and has served as the editors of Journal of Information Science and Engineering and International Journal of Ad Hoc and Ubiquitous Computing in 2004 and 2005, respectively. He has organized the 1st, 2nd and 3rd International Workshop on Peer-to-Peer Networked Virtual Environments in 2007, 2008 and 2009, respectively. His research interests include distributed algorithms for peer-to-peer networks, mobile ad hoc networks and wireless sensor networks. He is a member of ACM and IEEE.

Bing-Yu Chen (Member ’03) received the B.S. and M.S. degrees in Computer Science and Information Engineering from the National Taiwan University, Taipei, in 1995 and 1997, respectively, and received the Ph.D. degree in Information Science from The University of Tokyo, Japan, in 2003. He is currently 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 of the National Taiwan University, and also a visiting associate professor in the Department of Computer Science of The University of Tokyo. His research interests are mainly for computer graphics, human-computer interaction, and image and video processing. He is a member of ACM, ACM SIGGRAPH, Eurographics, IEEE, IEICE, and IICM.