Selection Strategies for Peer-to-Peer 3D Streaming

Wei-Lun Sung, Shun-Yun Hu, Jehn-Ruey Jiang

Department of Computer Science and Information Engineering
National Central University, Taiwan, R.O.C.

ABSTRACT

In multi-user networked virtual environments such as *Second Life*, 3D streaming techniques have been used to progressively download and render 3D objects and terrain, so that a full download or prior installation is not necessary. As existing client-server architectures may not scale easily, 3D streaming based on peer-to-peer (P2P) delivery is recently proposed to allow users to acquire 3D content from other users instead of the server. However, discovering the peers who possess relevant data and have enough bandwidth to answer data requests is non-trivial. A naive query-response approach thus may be inefficient and could incur unnecessary latency and message overhead. In this paper, we propose a peer selection strategy for P2P-based 3D streaming, where peers exchange information on content availability incrementally with neighbors. Requestors can thus discover suppliers quickly and avoid time-consuming queries. A multi-level area of interest (AOI) request is also adopted to avoid request contention due to concentrated requests. Simulation results show that our strategies achieve better system scalability and streaming performance than a naive query-response approach.

Categories and Subject Descriptors

I.3.2 [**Computer Graphics**]: Graphics Systems–Distributed / network graphics

Keywords

3D streaming, virtual environment, peer-to-peer

1. INTRODUCTION

In recent years, networked virtual environments (VEs) [19] have become more and more popular. Commercial VEs such as Massively Multiplayer Online Games (MMOGs) often contain large 3D content to present realistic, immersive environments for user interactions. Most MMOGs today need to be fully installed beforehand with CDs or DVDs for easier content access. However, as content gets larger and more dynamic, it would become increasingly inconvenient for new users to try, or for existing users to update the content. A better way thus is to acquire only the content of interest progressively via *3D streaming* techniques [20, 6, 3].

3D streaming refers to the continuous and real-time delivery of 3D content such as meshes, textures, animations and scene graphs over networks to allow user interactions without a full download [8]. A client renders the content according to certain data descriptions while downloading other data pieces concurrently. Similar to audio or video *media streaming*, the content needs to be fragmented into pieces before transmission. However, a major difference exists between 3D and media streaming: data is fragmented into sequentially ordered pieces and transmitted linearly in media streaming, whereas 3D content's transmission order is not fixed and requires individual visibility calculation based on the user's viewpoint and behavior [20, 13].

Existing 3D streaming schemes are based on the client-server architectures – all content is stored at a central server, and transmitted to clients upon requests [12]. However, as more clients join with more requests, the server becomes a bottleneck when the data requirement exceeds its capacity. To address this, peer-to-peer (P2P) delivery has been proposed to achieve better scalability [6]. As each peer has some data in its local cache, other clients could request from their peers first when in need of similar content. The system's serving capacity may thus improve dramatically. However, with a P2P approach, the issue of *peer selection* naturally arises. The purpose of peer selection is to choose suitable candidates so that content retrieval can be done quickly and efficiently. However, peer selection for P2P 3D streaming differs from file-sharing or media streaming due to data locality – peers who are nearby to each other in the virtual world see similar views and possess similar content. Suitable peers thus may be discovered with some predictability, but they also change constantly due to user movements. How to discover nearby neighbors and their data availability efficiently and continuously thus is the main challenge.

In this paper, we propose some new peer selection strategies for P2P 3D streaming. Peers periodically exchange incremental content availability with neighbors to allow requestors to learn of providers quickly, saving both message overhead and time. We also adopt a multi-level AOI request to avoid request contentions. In the rest of this paper: Section 2 provides background and related work; we describe our proposed methods in Section 3, and simulation evaluations in Section 4; conclusion is given in Section 5.

2. RELATED WORK

2.1 Peer-to-Peer VE

The earliest VEs connect users directly to each other, which only allows for limited users. The scalability is later improved by adopting the client-server architecture, where a central server keeps all data to limit the clients' bandwidth usage. However, the CPU and bandwidth resources of the server are still limited, which still impose bottlenecks for the system. As a result, P2P networks have been proposed to further improve scalability. The key idea behind a P2P-based VE (P2P-VE) is that, as users only need to interact within a limited *area of interest* (AOI) based on their current viewpoints, they only need to connect with the other users within the AOI. Due to user movements, the users within the AOI (called AOI neighbors) may change rapidly. How to discover the AOI neighbors efficiently thus is the central issue. Solipsis [10], SimMud [11], VON [7], and COVER [14] all try to provide neighbor discovery with the help of existing AOI neighbors or some superpeers, which could notify about new AOI neighbors without relying on any servers.

2.2 3D streaming

The goal of 3D streaming is to deliver 3D content in real-time, so that immediate interactions with the virtual world is possible. Although several factors (e.g., the rendering ability and bandwidth utilization) all affect the streaming quality, the most important limitation has been the bandwidth. Model simplifications and progressive transmissions have thus been used to deal with the limited bandwidth [20]. Even though content streaming requires additional bandwidth, it becomes less of a problem with better broadband networks. Prefetching techniques [4] additionally could reduce the potential delay users experience. In fact, VEs based on 3D streaming have already appeared (e.g., Second Life [15] and There.com), where terabytes of content are navigable via progressive transmissions. In the long run, install-then-play may no longer be practical or feasible, when the content becomes truly massive and dynamic (e.g., a virtual world with the scale of the globe[1]), or when VEs become as numerous as the websites today.

Current 3D streaming scheme may be classified into four types: object streaming, scene streaming, visualization streaming and image-based streaming [6]. In this paper we focus on scene streaming, which is an extension of object streaming where many objects constitute a scene. As the user's field of view (i.e., AOI) may not cover the entire VE, downloading the entire scene is not necessary. Hesina and Schmalstieg [18, 5] proposed that user can use a circular AOI, and keep the consistency of only the objects within AOI. Additionally, if users just rotate views but do not move, the objects within the AOI need not be re-acquired. Scene streaming may roughly be divided into two stages: object determination and object transmission [17]. The first stage determines which objects are visible given a user's view. The second stage decides the transmission order of objects based on visibility or the importance of objects. For example, when some objects are obscured by other objects, the parts which are visible may be transmitted first. Similar to other types of streaming, caching and prefetching are also important considerations [4].

[1]https://www.technologyreview.com/Infotech/18911/

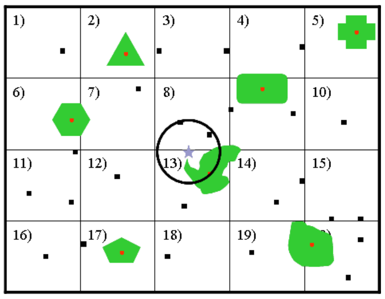

Figure 1: Design of FLoD.

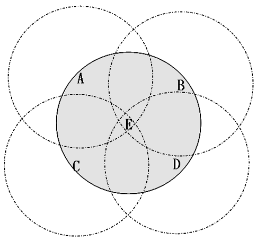

Figure 2: Example of request contention.

2.3 Peer selection in P2P 3D streaming

The purpose of peer selection is to choose the appropriate data providers so that good download performance is achieved. Hu et al. [8] propose a framework for 3D streaming on P2P networks called FLoD (see Figure 1, where star is a given user, dots are the other users. Objects are the various shapes). FLoD assumes that each 3D object can be fragmented into a *base piece* and many *refinement pieces*, and divides the virtual world into several cells, where each cell holds an individual *scene description* that describes the objects within the cell. When a peer's AOI overlaps with a new cell, it first tries to acquire the scene description, and then obtains the appropriate object pieces based on peer selection. FLoD implements a naive *query-response* method to discover content sources (i.e., which AOI neighbors have the desired pieces). To download data, peers first query their AOI neighbors, which are learned via a P2P-VE overlay such as VON [8]. Neighbors who possess the content would then respond positively, from which the requestors can then randomly choose a peer to issue requests. However, this approach has the following limits: 1) to prevent flooding, only limited query messages are sent at a time, which may produce few sources; 2) the overall download time is increased as queries are needed before requests; 3) random selection can cause a *request contention.*

Generally speaking, random peer selection achieves certain local load balancing as all AOI neighbors have equal chances to serve requests. However, from a global view, several requests may concentrate at an area and cause request contention. In Figure 2, if peer E is covered by the AOI of some other peers (A, B, C, and D). When these neighbors try to request data from within their AOI, peer E would get more requests on average due to the multiple AOI coverage. Therefore, when all peers are requesting, data providers near E may become overloaded. Moreover, if peer E itself is also a requester, it will get the worse performance because all of its potential providers could be serving other requestors.

Cavagna et al. [2, 16] have proposed another P2P 3D streaming scheme for urban scenes stored in a hierarchical and progressive tree structure called *Level of Detail Description Tree*, or LODDT. In this tree, each node represents a level of detail (LOD) for a certain region. The root node represents the roughest detail and also the whole environment. As the level increases, the details of the geometric data become more delicate and refined. Each node also has an *aura* representing a viewable area (i.e., similar to an AOI), of which if a peer enters, the appropriate LOD would be acquired by the peer. To facilitate peer selection, each peer informs its *Time to Serve* (TTS) to neighbors periodically. When peers enter an aura but do not possess the description data, they estimate what their neighbors might hold based on the neighbors' positions. With a list of potential candidates, a peer chooses the one with the least TTS to request. However, an obvious drawback is that, although requestors could select candidates inside the aura based on position info, some neighbors may be new and yet have any data to serve. Requests sent to these neighbors would be invalid and prolong the overall latency. Request contention may also occur as peers with the least TTS may receive more requests than others.

3. PROPOSED STRATEGIES

We observe that peer selection can be divided into two phases: *source discovery* and *source selection*. Source discovery is to find out which nodes might possess the desired data, and source selection is to choose a few nodes among potential candidates to request the actual data. Note that we assume that there are existing methods to allow each node to always connect with its AOI neighbors, and maintain them (i.e., discover new ones as nodes move) via a P2P-VE overlay such as VON[7, 8]. Direct message exchange among AOI neighbors is also possible. In this section, we describe our peer selection strategies according to these two phases.

3.1 Source discovery

We propose an active exchange method to discover data sources. Peers would periodically notify AOI neighbors about their own data availability incrementally. Besides an index number of message type, a notify message only contains the object ID (Obj_ID) and the *maximum continuous piece ID* (Max_PID) (Figure 3), which specifies up to which pieces have already been received (e.g., if pieces with IDs 1, 2, 4, 5 are received, 2 is the maximum continues piece ID). As peers can request data from multiple neighbors, the pieces received may be out of order. Here we assume that object re-constructions are strictly linear, so only the piece ID from the base piece (i.e., the first piece) to the maximum continuous piece are usable. Peers distribute the full availability information only to new AOI neighbors. For existing neighbors, only incremental updates are sent as new data pieces arrive. This way, peers spend less overhead to learn of the neighbors who might hold the desired data. Requestor can also easily create the list of request candidates based on the exchanged information, reducing the source discovery time.

Additionally, as non-AOI neighbors might also possess relevant data, a peer would keep its neighbors' data availability information even after they move out of AOI, and would still exchange the neighbors' positions until they are beyond a specified distance. In Figure 4 (solid circles are the original AOI and dotted circles are the *extended candidate buffer*),

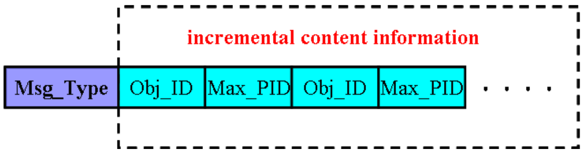

Figure 3: Format for an availability update.

we assume that peer B already has the scene description and the object data. When both A and B move into each other's AOI, they would learn of each other's data availability. After B moves out of A's AOI, peer A would still keep B's information. So when A needs to request new objects, it can add B into the candidate list as B is still within the extended buffer. The potential data sources can thus be increased with few overheads. These peers additionally facilitate content dissemination, as with more providers, the load will be shared by more peers.

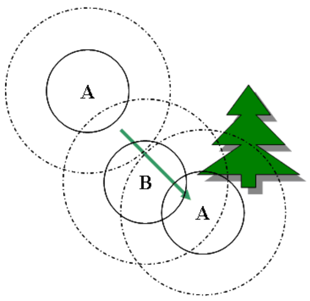

Figure 4: Extended candidate buffer.

3.2 Source selection

Once we have a list of candidate sources, we need to select the suitable ones for efficient retrieval. Although random selection is quite simple and achieves local load balancing, it may cause request contention when multiple nearby peers are requesting from the same data. In order to reduce contention, we propose a *multi-level AOI request* for source selection.

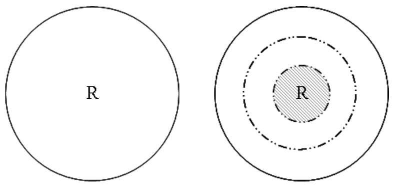

Figure 5: Original vs. multi-level AOI request: (L) Original AOI (R) Multi-level AOI, the shaded area has the highest request priority.

The idea for multi-level AOI is to divide the original AOI into several concentric *levels*, where each level closer to the center (in terms of virtual coordinates) has a higher request priority than those further out (Figure 5). Peers within the higher levels are requested first. If there is more than one candidate at the same level, the requestor then chooses randomly. The central concept is to concentrate data requests closer to the requestor itself, so that contention is avoided among peers who desire the same data. For instance, if all five requestors in Figure 6 request from the higher priority area first, their request areas then would not overlap. Providers thus serve in a more balanced manner to reduce contention and enhance performance. However, multi-level AOI request may extend request times slightly. As peers now request from fewer providers for the same amount of data, more time may be needed to fulfill each requests.

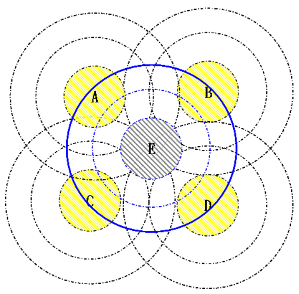

Figure 6: Example of multi-level AOI request.

4. SIMULATION EVALUATION

4.1 Simulation setup

We use simulations to evaluate our proposed selection strategies. Our simulation is based on FLoD [8], as FLoD uses a simpler 2D plane to organize the objects, and its source code is publicly available [1]. Existing P2P streaming approaches focus mainly on sequential data, and do not fully support the requirements of interactive 3D applications. As such, they are not directly comparable with our approach. Our main goal thus is to compare the proposed selection strategies with those used by the original FLoD. In order to simplify the naming for different strategies, we use ”QR” and ”EE” to represent the source discovery strategies ”query-response” and ”exchanged & extended candidate buffer”; ”Rand” and ”ML” for the source selection strategies ”random selection” and ”multi-level request”. So for example, ”QR-Rand” is the naive query-response and random selection used by the original FLoD.

To initialize the simulation, 3D objects are randomly deployed in the virtual world. As in FLoD, we assume that each 3D object can be fragmented into a *base piece* and several *refinement pieces* [8], where the base piece gives a rough outline of the object, and refinements restore the object to its original shape. We adopt similar simulation parameters as used in FLoD, which is based on the data from a game prototype. Each object is 15KB (3KB is the base piece, and refinements are 1.2KB each). Since the number of objects in each cell is not necessarily equal, the sizes for scene descriptions are not fixed, but are roughly about 1.7KB each.

In the simulation, a number of nodes are first created to represent virtual world users. Each node adopts *random way-point* movements at a fixed speed. The initial node positions are also randomly placed. When the simulation starts, all nodes stay at the starting position until the average possessed data of each peer reaches 80% of those within their view. This ensures that when the nodes start moving, they already have partial data to serve other nodes. Moreover, in order to simulate a more realistic environment, we limit the node bandwidth to be 1 Mbps download and

Table 1: Simulation parameters

World dimension (units)	1000x1000
Cell size (units)	100x100
AOI-radius (units)	75
Time-steps	3000
Number of nodes	50 - 500 (in 50 increment)
Number of objects	500
Node speed (units / step)	1
Client cache size (KB)	20 (scene descriptions) 700 (object pieces)

256 Kbps upload. The central server’s bandwidth is limited at 10 Mbps for both download and upload. The simulation proceeds for 3000 *time-steps* where each step is 100 ms each. For simplicity, constant latency is assumed between all nodes (i.e., each message sent is received by a neighbor in the next time-step, unless delayed due to bandwidth limits). For the query-response strategy, 10 query messages are sent to neighbors every 5 time-steps. For our proposed scheme, the extended candidate buffer radius is 150 (inclusive of the original AOI), and the number of AOI levels is 4. We simulate different scenarios with an increasing number of nodes within the VE, to see how the system may scale. Specific parameters for our simulations are shown in Table 1.

4.2 Simulation results

We collect the statistics from the last 1000 steps for our analysis, as the system needs some time to achieve the steady state which we are interested in. We will discuss the simulation results in respect to the streaming quality and system scalability below. Note that node size in the following figures indicates the total number of peers in the system.

4.2.1 Streaming quality

We first show our results on streaming quality, which is quantified by the following metrics used in FLoD [8]: hit ratio, base latency, and fill ratio:

* **Hit ratio:** Ratio between successful requests and the total number of requests sent.

* **Base latency:** Duration between the initial request and the time the base piece is acquired.

* **Fill ratio:** Percentage of the possessed data among the required data for a view.

Figure 7 shows the hit ratio, where we can see that with random peer selection, hit ratio drops when the number of total users (hence the user density in the fixed-size area) increases. This is because random selection causes resource contention and increases the possibility to request from peers without serving capacities. Besides, we also see that using ”exchanged & extended candidate buffer” would decrease the hit ratio. It is because peers only notify content availability to their neighbors. However, peers may remove old content when the cache is full, yet, as such removal is never communicated to neighbors to conserve bandwidth, peers may end up having a higher probability to request data from unavailable providers.

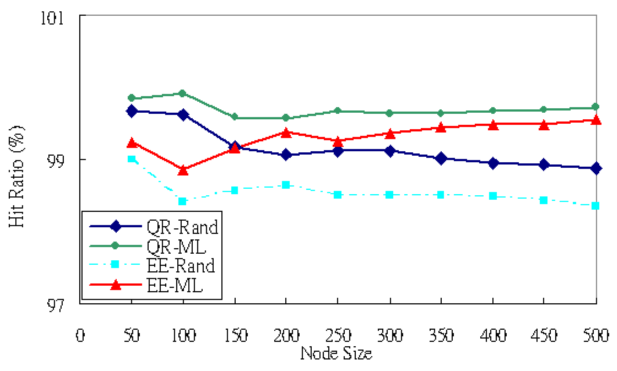

Figure 7: Hit ratio.

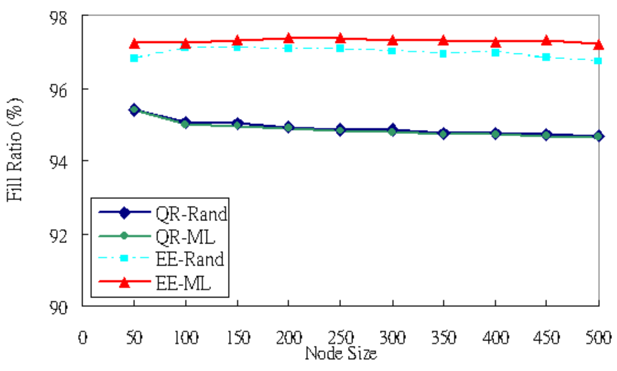

Figure 9: Overall fill ratio.

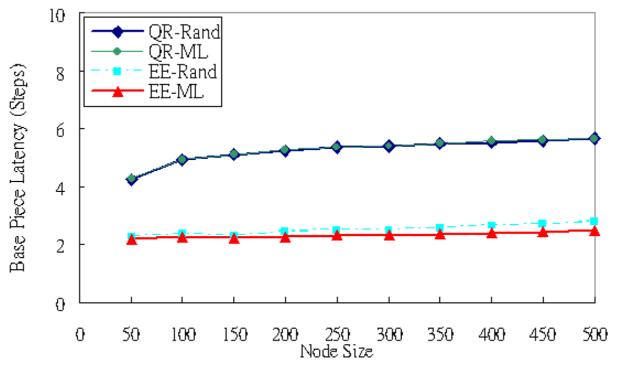

Figure 8: Base latency.

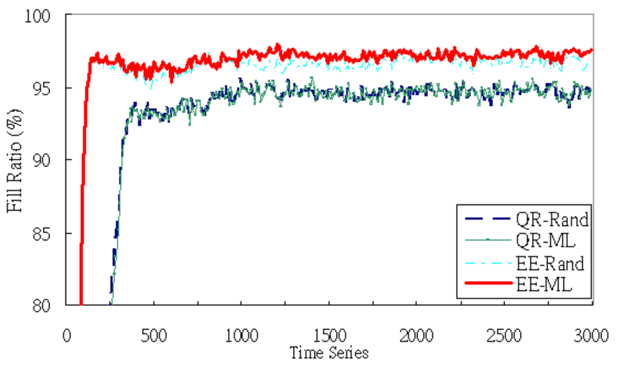

Figure 10: Overall fill ratio time series.

Figure 8 shows the latency to acquire the base piece, which indicates how fast the outline of an object can be shown once it becomes visible. We can obviously see that the original strategies used by FLoD need longer time to acquire the base pieces, in order to wait for the response from the data providers. With our discovery strategies, peers may reduce the base latency by almost half (i.e., it takes only a round-trip of 200ms to see the base piece). Multi-level requests also help to reduce latency, because it decreases the probability for request contention to occur, and enhances the hit ratio of peers, needless re-requests thus are avoided.

From Figure 7 and 8 we already see that our methods achieve better performance when the number of peers increases. Figure 9 shows the overall fill ratio, which further validates these results, where the fill ratios are generally higher than the original FLoD strategies. Moreover, Figure 10 shows the time series of the overall fill ratio. When the peers start moving, the overall fill ratio first drops then restores to a steady state. We can see that the naive strategies used by FLoD requires a longer time to achieve the steady state, which shows that the original peer selection is slower in disseminating data. A possible reason is that in the original selection, peers who receive queries do not necessarily possess the data. So there are fewer actual providers than the number of query targets. The requestors are thus unable to acquire the desired data effectively.

4.2.2 *System scalability*

Scalability describes whether a system is flexible enough to allow users to join simultaneously. Note that although we mainly discuss about the number of total users in the system (or *system scalbility*), and not the scalability of users within the AOI (or *AOI scalability* [9]), the two are directly related in our simulations as the world size is fixed (i.e., the increase in total number of users directly relates to an increase in user density). We can judge whether a system is scalable by considering its resource consumption. In a P2P system, the main limiting resource is the bandwidth used by both the server and the clients [7]. The system can scale as long as bandwidth consumptions stay below the capacity limits of all nodes. Figure 11 shows the server's bandwidth consumption. We can clearly see that the peer selection in the original FLoD consumes more bandwidth than our proposed strategies due to longer latency and the lack of data sources. When peers need more time to inquire for data, there would be fewer nodes with available data to serve. When the limited data sources cannot serve, the only way for peers to acquire data is to request from the server. Additionally, the original FLoD limits the number of query messages at a time, in order to avoid request congestion, but it also makes the peers to be aware of fewer candidates. The peers who possess the data but do not receive queries thus will be unable to serve. As data is disseminated slowly, the central server will thus get more requests and consume more bandwidth.

Figure 12 shows the average source discovery bandwidth of peers, which is the main overhead of using a P2P ap-

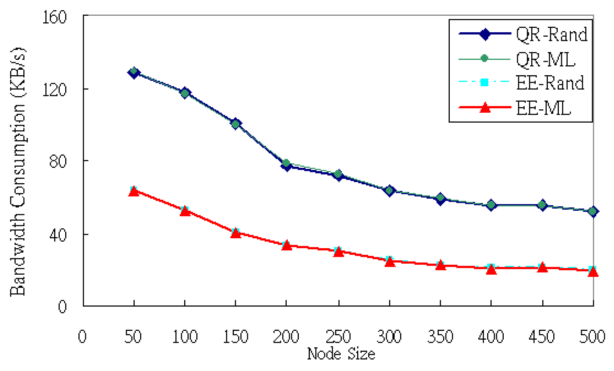

Figure 11: Server bandwidth consumption.

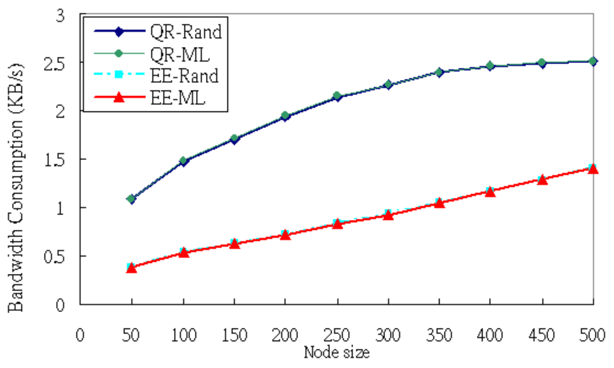

Figure 12: Bandwidth used for source discovery.

proach to download content as compared with client-server. It shows that our scheme also uses less bandwidth than in the original FLoD. As peers notify neighbors the piece information incrementally, only a small amount of bandwidth is required. Peers thus have more bandwidth to serve the neighbors, which in turn helps to enhance overall system performance.

5. CONCLUSION

In this paper, we propose some new peer selection strategies for P2P-based 3D streaming VE applications that improve both the streaming quality and the system scalability. By exchanging data availability information incrementally, peers are aware of potential data providers immediately, thus reducing the time to request, and bandwidth for inquiry. We also adopt a multi-level request mechanism for peers to concentrate data requests closer to themselves, so that request contention is reduced. As shown by simulations, our proposed strategies indeed improve both system scalability and performance.

The ideas for P2P 3D streaming is also usable for other spatially interactive content streaming. For example, the real-time streaming for 3D virtual globes is a potential application beyond the streaming of game content. For future work, we plan to consider balancing the requests for all peers more equally, and utilizing pre-fetching techniques to further reduce the time to retrieve new 3D objects. Consideration of the physical topology is also an issue, as peers close in the virtual space may be far apart on the physical networks. Applicability of our proposed strategies to other P2P 3D streaming schemes is another avenue for investigations.

6. REFERENCES

[1] Ascend project. http://ascend.sourceforge.net, 2008.

[2] R. Cavagna, C. Bouville, and J. Royan. P2p network for very large virtual environment. In *Proc. VRST*, pages 269–276, 2006.

[3] W. Cheng et al. An analytical model for progressive mesh streaming. In *Proc. ACM Multimedia*, 2007.

[4] J. Chim et al. Cyberwalk: A web-based distributed virtual walkthrough environment. *IEEE TMM*, 5(4):503–515, 2003.

[5] G. Hesina and D. Schmalstieg. A network architecture for remote rendering. In *Proc. DIS-RT*, page 88, 1998.

[6] S.-Y. Hu. A case for 3d streaming on peer-to-peer networks. In *Proc. Web3D*, pages 57–63, 2006.

[7] S.-Y. Hu, J.-F. Chen, and T.-H. Chen. Von: A scalable peer-to-peer network for virtual environments. *IEEE Network*, 20(4):22–31, 2006.

[8] S.-Y. Hu et al. Flod: A framework for peer-to-peer 3d streaming. In *Proc. INFOCOM*, 2008.

[9] J.-R. Jiang, Y.-L. Huang, and S.-Y. Hu. Scalable aoi-cast for peer-to-peer networked virtual environments. In *Proc. CDS*, 2008.

[10] J. Keller and G. Simon. Solipsis: A massively multi-participant virtual world. In *Proc. PDPTA 03*, pages 262–268, 2003.

[11] B. Knutsson, H. Lu, W. Xu, and B. Hopkins. Peer-to-peer support for massively multiplayer games. In *Proc. INFOCOM*, pages 96–107, 2004.

[12] F. W. B. Li, R. W. H. Lau, and D. Kilis. Gameod: an internet based game-on-demand framework. In *Proc. ACM VRST*, pages 129–136, 2004.

[13] K. Mayer-Patel and D. Gotz. Adaptive streaming for nonlinear media. *IEEE Multimedia*, 14(3):68–83, 2007.

[14] P. Morillo et al. *Providing Full Awareness to Distributed Virtual Environments Based on Peer-To-Peer Architectures*. June 2006.

[15] P. Rosedale and C. Ondrejka. Enabling player-created online worlds with grid computing and streaming. Gamasutra Resource Guide, 2003.

[16] J. Royan, P. Gioia, R. Cavagna, and C. Bouville. Network-based visualization of 3d landscapes and city models. *IEEE CG&A*, 27(6):70–79, 2007.

[17] J. Sahm, I. Soetebier, and H. Birthelmer. Efficient representation and streaming of 3d scenes. *C & G*, 28(1):15–24, 2004.

[18] D. Schmalstieg and M. Gervautz. Demand-driven geometry transmission for distributed virtual environments. *CGF (EG 1996)*, 15(3):421–433, 1996.

[19] S. Singhal and M. Zyda. *Networked Virtual Environments: Design and Implementation*. ACM Press, 1999.

[20] E. Teler and D. Lischinski. Streaming of complex 3d scenes for remote walkthroughs. *CGF (EG 2001)*, 20(3), 2001.