Discovery of Physical Neighbors for P2P 3D Streaming

Chang-Hua Wu, Shun-Yun Hu, Li-Ming Tseng
Department of Computer Science and Information Engineering
National Central University, Taiwan, R.O.C.
johnny@dslab.csie.ncu.edu.tw, syhu@csie.ncu.edu.tw, tsenglm@cc.ncu.edu.tw

***Abstract*— Many peer-to-peer-based virtual environment (P2P-VE) solutions have been proposed recently to improve the scalability of distributed virtual environment (VE) systems. By organizing peers to realize neighbor discovery through mutual peer collaboration, P2P-VEs allow systems to scale without additional server provision. Related services such as the real-time delivery of 3D content (i.e., 3D streaming) can also be supported with P2P approaches. However, the selection of peer connections based purely on logical neighbors in the virtual world may cause serious latencies. Accordingly, we propose *Discovery of Physical Neighbor* (DPN) that groups peers with physical proximity together to reduce the latencies among peer communications. The maintenance of DPN is distributed among the peers via cooperation. We also evaluate our approach on P2P-based 3D streaming via simulations to verify DPN's effectiveness**

Keywords-networked virtual environment, peer-to-peer network, 3D streaming, locality-aware, landmark.

I. INTRODUCTION

Networked virtual environments (VEs) [1] are Internet applications that combine virtual reality with networking to provide immersive experiences to people. VEs were developed as military simulators in the 1980s and have evolved to Massively Multiplayer Online Games (MMOGs) in the mid-1990s. Today, MMOGs are the most popular form of VEs and have become a multi-billion dollar industry. In order to build graphically realistic VEs, most MMOGs contain a lot of 3D content (e.g., the most popular *World of Warcraft* is over 5 GB). Existing MMOGs predominantly require users to pre-install the content. However, as content size grows into the range of terabytes (e.g., the social MMOG *Second Life* has over 34 TB of content in 2007), pre-installation will no longer be a viable approach. The real-time streaming of 3D content (or *3D streaming*) thus may become important in the near future. As more people are attracted to VEs, the rapid growth of large-scale VEs has introduced serious challenges to existing network architectures. Bringing additional resources to the system with each user joined is the advantage of *peer-to-peer* (P2P) architectures as an alternative solution that may solve the scalability problem of large-scale VEs.

P2P architecture differs from client-server architecture in that the basic units are commodity PCs with equivalent functionality. Due to P2P's scalability and affordability, many P2P-based VE (P2P-VE) designs are recently proposed [2], solving problems from neighbor discovery to content delivery. However, when peers or neighbors in the P2P network are selected without considering physical distances, serious latency may result for the system. Considering the characteristics of P2P-VEs, we thus propose a method called *Discovery of Physical Neighbor* (DPN). We separate the peers into several groups according to physical proximity, and then maintain each group with a tree architecture formed by user nodes to alleviate the loading of the server. We evaluate our design in the support of P2P-based 3D streaming, as it is both volume-intensive and latency-sensitive. To get the content as soon as possible, we also design priorities for content delivery, and evaluate our method via simulations. The rest of the paper is organized as follows: Section II discusses related work, and we present DPN in Section III. To evaluate DPN, Section IV describes the related simulations. Finally, conclusions are given in Section V.

II. RELATED WORK

A. P2P-VE and 3D streaming

VEs are virtual worlds where each user interacts with other human or computer players with a virtual identity called *avatar*. Similar to the real world, participants may perform different actions such as moving to new locations, looking around at the surroundings, using items to perform tasks, etc. In the real world, although many users and events may exist, each user is only affected by nearby users or events. Thus, we can exploit this fact with a fundamental concept called *area of interest* (AOI) in a generalized VE. Users can be conceptually seen as coordinate points on a 2D plane (as *nodes*), each with a limited visibility range. Because each node's AOI is limited, the desired interaction or content is localized and easily identified. The basic premise for P2P-VEs thus is that given a node's position and AOI, all neighboring nodes within the AOI can be known for interactions. If these *AOI neighbors* can be properly discovered without the server's involvement, the system can scale and provide various additional services (e.g., state management [3] and content management [4]).

The main problem in content management is to allow each user see the 3D content within the AOI in a timely manner. While *download-and-play* can easily achieve this goal, users may prefer to enter the VE quickly without waiting. *3D streaming* [5] techniques thus have been proposed for the progressive transmission of content. Similar to audio or video media streaming [6], 3D content needs to be fragmented into pieces at a server, before it can be transmitted, reconstructed, and displayed at the clients. But unlike media streaming, as user interests depend on the positions and paths in the VE, the streaming sequences vary from user to user and require individualized visibility calculations. FLoD [7] describes the first framework for P2P 3D streaming [4], where by utilizing the often overlapped visibility among AOI neighbors, if content is requested from the neighbors first, the server's workload may be much relieved. To properly discover and maintain the neighbors, FLoD utilizes a P2P overlay called VON [8].

To help the clients learn of the objects in view, the world is first partitioned into square cells [9]. Each cell has a small *scene description* that specifies the objects within. Each 3D object is specified by a unique ID, location point, orientation and scale within the scene description. Determining the visible objects can thus be done in a fully distributed manner, as each node is able to locally determine the cells covered by its AOI. In Figure 1, the red circle is the AOI of the star node, and triangles are other user nodes. Various shapes are 3D objects, with their location points as dots. When entering a new area, a client first prepares a scene request to obtain scene descriptions from its AOI neighbors or the server. Data pieces for AOI objects are then requested. This process of content request is repeated iteratively as a client moves around. Note that for our discussions, each node is only concerned with getting the 3D objects for display. The management of other object states (e.g., health and experience points of an avatar) is beyond our scope.

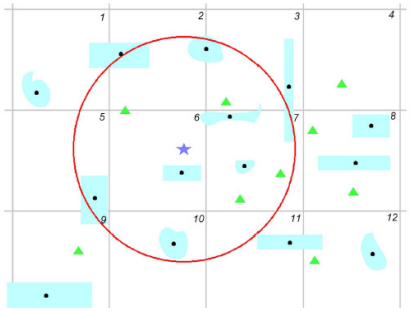


Figure 1. The world model of FLoD [7]

B. *Locality-Awareness*

In the context of P2P-VE systems, we refer to *locality awareness* as considering physical distances among the peers in terms of latencies. Two popular ways to measure latency in networks are hop count and round-trip time (RTT). Because geographic proximities (i.e., hop count) sometimes do not equal to actual delays (i.e., network distances), and RTT considers both network and processing delay, we will adopt RTT as our main distance metric. In practice, it is also more efficient to use RTT to estimate the latencies between end-to-end nodes [10].

Several methods for RTT estimations exist. Hsieh's method [11] finds nearest neighbors in a replication-aware P2P-CDN by binning. Binning [12] is an approach where nodes partition themselves into bins such that the nodes in a given bin are relatively close to one another in terms of network latency. A distributed binning scheme can be incorporated into distributed systems such as P2P networks and content distribution systems. Binning requires a set of well known *landmark* machines spread across the Internet. An application node measures its distance by RTT to the landmarks and independently selects a particular bin based on these measurements. A node's RTT measurements to each landmark offer two kinds of information: 1) the relative distance of different landmarks from the given node and 2) the absolute values of these distances. Figure 2 shows how the distances are measured with a specific example:

The relative distance: we try to augment the landmark ordering of a node with a level vector, where each landmark has a corresponding level number. For example, if Node A's distances to landmarks L_1, L_2, and L_3 are 232ms, 51ms and 117ms respectively. Its ordering of the landmarks is "$L_2L_3L_1$".

The absolute distance: Here the range of possible latencies are divided into 3 levels; level 0 for latencies between [0,100]ms; level 1 for latencies between [100,200]ms; and level 2 for latencies greater than 200ms. Using the 3rd levels, node A's level vector corresponding to its landmarks ordering is "0 1 2".

Thus, after combining the relative with the absolute distances, node A's bin is "$L_2L_3L_1$: 0 1 2".

The above binning scheme thus does a reasonable job of placing nearby nodes into the same bin, (e.g., node C and D). In other words, node C and node D represent a set of approximate physical neighbors. Binning is simple and requires very little from measurement and infrastructure. It is also very scalable because nodes can independently discover their bins without communicating with other nodes.

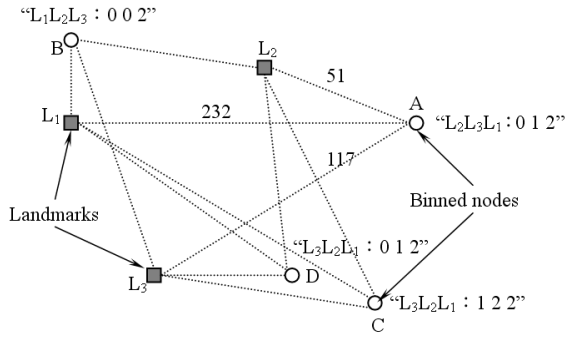


Figure 2. Distributed binning [12]

III. DISCORVERY OF PHYSICAL NEIGHBORS

The apparent problem with a basic design such as FLoD is that neighbor formations are purely based on their logical relationship but without knowledge of the physical networks. The transmission thus may incur unacceptable delays for the users. We assume that it would be more efficient if the data is received from not just the virtual neighbors, but also nearby physical neighbors. Furthermore, for users just walking around,

9781-4244-3941-6/09/$25.00 ©2009 IEEE

often only a glimpse of the surroundings (and not details) is needed. The priority in selecting neighbors thus is speed over completeness. As finding the absolute nearest neighbors is unnecessary (especially if the process is slow), we thus adopt a metric to discover physical neighbors only approximately.

A. System Model

Our goal is to find close physical neighbors for content exchange. We first outline the main system components and their functions, composed of *content servers* and *peers*. Content servers are the initial hosts of all content and respond to content requests from peers. Content servers also perform landmark tasks by allocating new peers to a physical group and adjust the tree structure that maintains each group. Peers make requests to both neighboring peers and/or the server. They may also accept requests from other peers and serve them. To find neighbors and exchange content, each peer maintains its AOI neighbors via a P2P overlay. To manage its membership within the tree structure of a group, each peer maintains a grandfather, father and child node table (explained later).

Figure 3 shows the overview of our system model. The upper part is a diagram of the virtual world in FLoD, which shows the partition of the scenes, the position of objects, and the peers in the VE. The lower part is the grouping with DPN. All peers are separated into one of the several groups with short latencies among members. The arrows show mappings of the same peer from the VE to the real world.

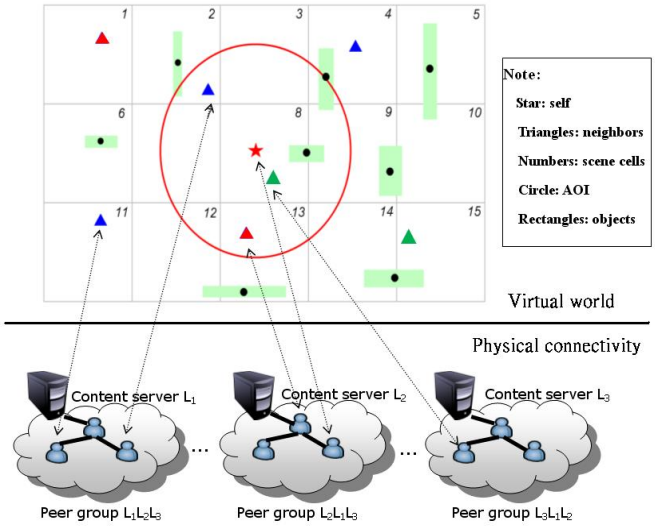


Figure 3. System overview

B. DPN Algorithm

1) Group creation

In this section, we introduce the process of group creation. First, the locations of landmarks must be determined. How many and how distributed are two important issues. Several methods to pick suitable landmarks exist [11]. Most assume that there are m servers, and n servers are picked among m servers to become landmarks ($m \geq n$). To simplify the process, we set the content servers as landmarks directly and assume that they are well-distributed. All peers then divide into $n!$ groups at most for n content servers. Each user will obtain RTT with all landmarks by pinging. After collecting and sorting the RTT, the servers will assign each user to a group by binning.

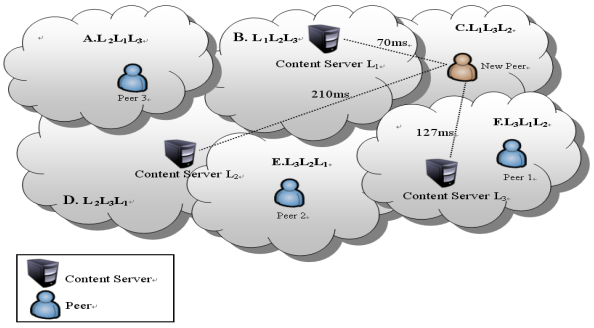


Figure 4. The configuration of grouping

As shown in Figure 4, if a new peer joins, and servers L_1, L_2, and L_3 are landmarks, the RTT results can be separated to six possible groups: A: $L_2L_1L_3$, B: $L_1L_2L_3$, C: $L_1L_3L_2$, D: $L_2L_3L_1$, E: $L_3L_2L_1$, and F: $L_3L_1L_2$. The RTT between a new peer R_1 and all three landmarks are RTT(R_1, L_1): 70ms, RTT(R_1, L_2): 210ms, and RTT(R_1, L_3): 127ms. The content server then sorts the results and determines the relative distance $L_1L_3L_2$ for the new peer. Hence, the new peer will be assigned to group C: $L_1L_3L_2$. This way, we can partition all peers to several groups automatically.

2) Group preservation

Once a group is successfully created, we need to maintain each group with procedures that include joining, leaving and maintaining. After binning by the server, a new peer performs the following join procedure (Figure 5).

- According to the new peer's relative distance, the content server sends the new peer to its peer group by notifying the new peer of an existing node. (Step 1).
- The peer sends the connection request to this existing node for connecting into an AVL-tree. (Step 2).
- The new peer creates the grandfather, father, and child nodes table, and exchange information with these nodes to update the table, while learning other potential father nodes from the grandfather (Step 3).

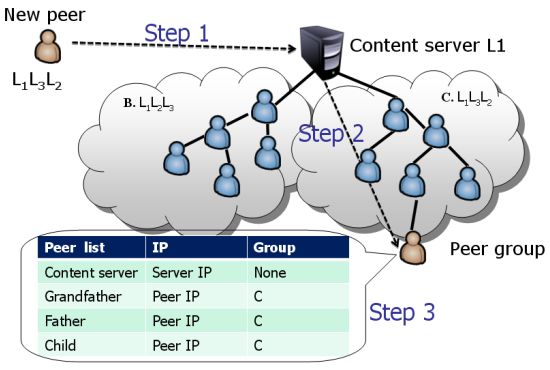
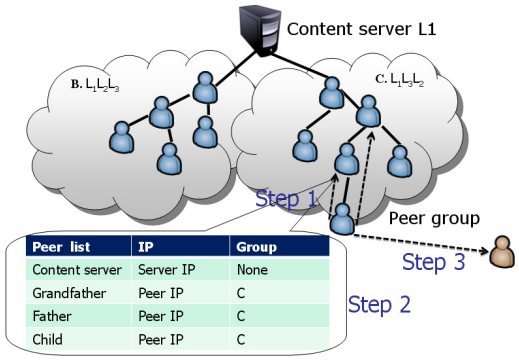
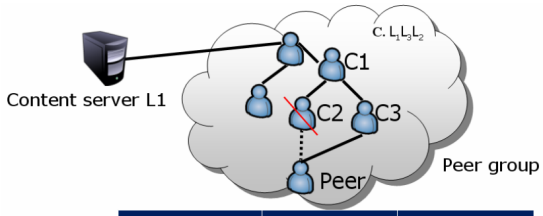


Figure 5. The join procedure



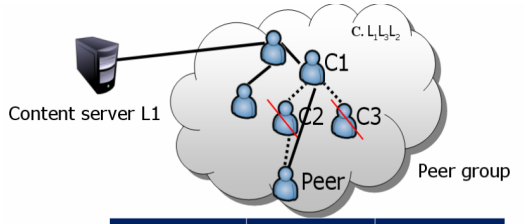
Peer list	IP	Group
Content server	Server IP	None
Grandfather	Peer IP	C
Father	Peer IP	C
Child	Peer IP	C

Figure 6. The leave procedure



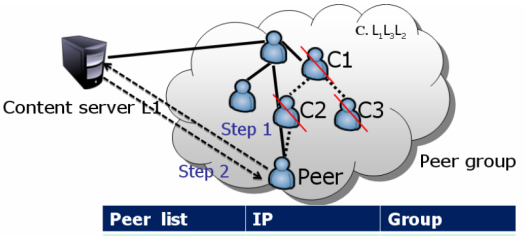
Peer list	IP	Group
Content server	Server	None
Grandfather	Peer C1	C
Father 1	Peer C2	C
Father 2	Peer C3	C

Figure 7. The maintenance procedure A



Peer list	IP	Group
Content server	Server	None
Grandfather	Peer C1	C
Father 1	Peer C2	C
Father 2	Peer C3	C

Figure 8. The maintenance procedure B



Peer list	IP	Group
Content server	Server	None
Grandfather	Peer C1	C
Father 1	Peer C2	C
Father 2	Peer C3	C

Figure 9. The maintenance procedure C

For proper leaving (Figure 6), such as logout, a leaving peer notifies the nodes in its table, then disconnects from its group.

- The leaving peer notifies the nodes in the table (Step 1).
- The notified nodes modify their tables (Step 2).
- The leaving peer leaves normally (Step 3).
- Content server adjusts the AVL-tree structure.

Lastly, we describe the basic procedure for maintenance. In each group, every peer pings its direct father node periodically within a period *t*. When the node is alive, the tree structure is correct. If the direct father does not reply, we trigger one of the three maintenance procedures listed below. Afterwards, the cleanup procedure is used.

a) Invalid direct father: (Figure 7)

- The peer pings its direct father C2 periodically.
- If the father does not respond, the peer looks for the other father C3 in the table and requests connection.
- After receiving the reply, the peer connects to C3.

b) Invalid all fathers: (Figure 8)

- If the checking procedure *a)* fails, where all other father nodes (e.g., C3) also do not respond.
- The peer requests connection with the grandfather C1.
- After receiving reply, the peer connects to C1.

c) Invalid father nodes and grandfather node: (Figure 9)

- If the procedures *a)* and *b)* both fail.
- The peer sends connection query to the content server.
- The server replies with an existing node to the peer.
- The peer requests connection to this existing node.
- After receiving the reply, the peer connects to this node.

d) Cleanup

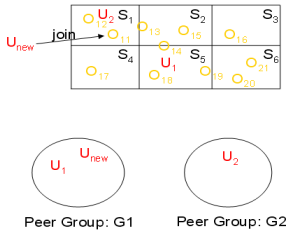
- The peer, the content server, and its new direct father exchange information to update the table.
- Content server adjusts the AVL-tree structure.

C. Neighbor selection

When a user walks in the scene, the peer can discover AOI neighbors from the P2P overlay. When selecting neighbors to retrieve desirable content, both logical AOI neighbors and physical DPN neighbors can be considered. The concept of nearby neighbor thus is enhanced by considering both aspects. Below we show how this neighbor selection is done.

1) Selection flow

Figure 10 shows the schematic and the pseudo-code for selecting neighbors to perform content request. We define an N-tuple <G1, G2, G3, …, Gn> as each peer group in the real world; the scene S = < S_1, S_2, S_3, S_4, S_5, S_6>; the object O = <O_{11}, O_{12}, …, O_{21}>; and the user node U = <U_1, U_2, U_{new}> in the virtual world, where U_{new} is a new peer. Finally, T_k is the current logical time and T_{k-1} is the previous logical time (in the form of a game *tick* or *time-step*). We assume that a globally consistent logical time exists for nodes regardless inside or outside the AOI.



Pseudo code:
U_{new}, U_1: G1 ; //U_{new} and U_1 are in the same group G1
U_2: G2 ; //U_2 is in the other group G2

while U_{new} joins a scene S (e.g., S_1)
if U_1 in scene S and owns objects (e.g., O_{11},O_{12}, O_{13}) in time T_k
request data transfer from U_1
else if U_1 owns the object O in previous time T_{k-1}
request data transfer from U_1
else if U_2 in scene S owns the object O in current time T_k
request data transfer from U_2
else
request data transfer from content server

Figure 10. Flow of neighbor selection

2) Data streaming

In order to receive the content as soon as possible, we classify the streaming priority as shown in Table I. Due to latencies, physical neighbors are more important than logical neighbors. Peers in the same group thus indicate shorter latency and must be chosen with the higher priority.

TABLE I. THE STREAMING PRIORITY

Priority	Group	AOI
1	The same	Inside
2	The same	Not inside
3	Different	Inside
4	Server	Server
5	Different	Not inside

In Figure 11, a peer will request data with priorities ranging from 1 to 5 (except 4 for the server), and G1, G2, and G3 specify the physical groupings for peers. We assume that even if a neighbor is far logically, the neighbor may still own some desired content and thus can be a potential source.

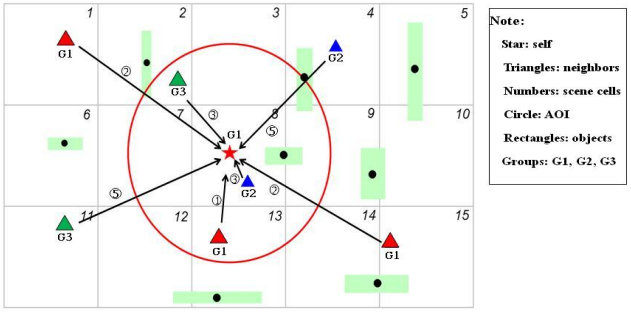


Figure 11. Priority of data streaming

IV. EVALUTION

As it is difficult to perform real experiments with many peers, we verify DPN via the same discrete-time simulator used by FLoD [7]. One decision to make is how many physical groups should be created. Too many or too few will obviously affect performance. As the size depends on the effectiveness of landmarks, the decision is also a landmark selection problem. The efficiency of landmark-based binning [12] depends on:

- The type and size of the topology the landmarks reside.
- The connection size among nodes.
- The number of selected landmarks.

Hsieh et al. [11] have simulated the above conditions in a previous work and report that when the number of landmarks increases from 5 to 6, the RTT is reduced most significantly at 38%. The decrease in RTT then would not improve with more landmarks. Thus, we choose to simulate only three landmarks.

A. Simulation environment

The FLoD simulator runs on top the P2P-VE overlay VON [8]. To set up a VE, a number of objects are randomly placed on a 2D map partitioned into square cells. For simplicity, we assume that each object has only one set of data pieces. A number of nodes are put randomly in the virtual environment, and stay at their joining locations until the system's average *fill ratio* exceeds 99%. The fill ratio [7] is defined by the ratio of data volumes between the client's obtained data and visible data. This is to give each node an initial set of data to share. The nodes then move with constant speeds, and request scene descriptions or data pieces as needed. The simulator runs at time-steps of 100ms each. In order to simulate a home ADSL environment, the bandwidth limits are set as 1 Mbps download and 256 Kbps upload for typical broadband clients and a 10 Mbps symmetric connection for the server. Each object is set to 15 KB, where the base piece is 3 KB and 10 refinement pieces are 1.2 KB each. Scene descriptions are around 300 to 500 bytes each.

The latencies among the nodes are randomly assigned between 1 to 15 time-steps (i.e., 100ms to 1500ms). Each simulation proceeds for 3000 steps, which is equivalent to 300 seconds assuming 100ms per step. As we are interested in the steady-state behavior, we collect statistics only for the last 2000 steps in each simulation. Finally, the max number of groups is 3! = 6 due to the use of 3 landmarks. Table II shows the more detailed parameters.

TABLE II. SIMULATION PARAMENTS

World dimension (units)	1000 x 1000
Cell Size (units)	100 x 100
AOI-radius (units)	75
Time-steps	3000
Number of nodes	50 - 500
Number of objects	500
Node speed (units / step)	1
Latency (time-steps / random)	1 - 15

B. Analysis

1) Base latency

We define the time between the initial query and the time the first piece becomes available at a client as *base latency* [7]. It serves as an indicator of how soon a user can start to navigate in a new scene. In Figure 12, we can clearly see that if the number of nodes is few, the latencies are larger for both nodes with and without DPN. As the number of nodes increases, the latencies will decrease and approach a constant number. We see that base latency without DPN is always higher. Note that for comparison, we also add an *optimal* line where the latencies among all nodes are always constant at 100ms (i.e., one time-step), which shows the optimal streaming performance.

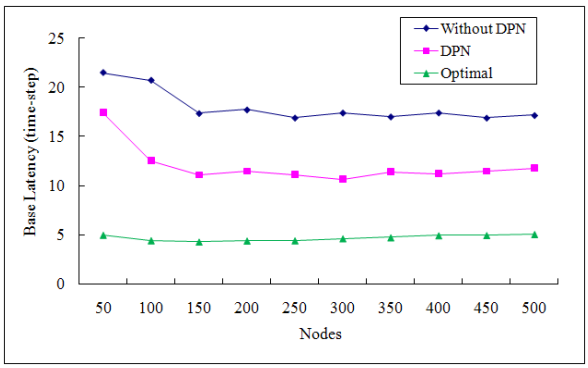


Figure 12. Base Latency

2) Server request ratio

Server's loading is another important metric that can be defined by the ratio between the content retrieved from the server and all the requested content. It also indirectly measures the system's scalability (i.e., whether the server will overload as user size increases). As the number of peer increases, the system will likely operate better if the ratio is kept low. As shown in Figure 13, the server request ratio for objects is higher when nodes are few. However, as more nodes join, the server request ratio reduces more. So if the number of the nodes is large enough, the loading on server is light no matter whether DPN is applied. In other words, the application of DPN will not burden the system too much. The simulation results show that scalability of the system is still maintained well with DPN. Note again that with a constantly low latency, optimal performance can reach a server request ratio of 1%.

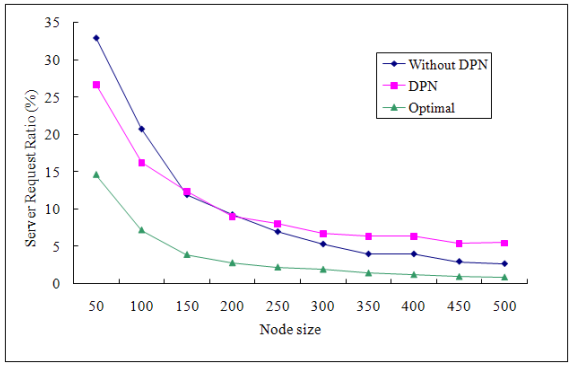


Figure 13. Server Request Ratio - Object

3) Transmission size

In Figure 14, we can see the bandwidth usage of nodes with DPN is higher than those without using DPN. The transmission size grows faster when using DPN, which indicates that the bandwidth utilization of peers with DPN is better than those not using DPN, under the same bandwidth constrain.

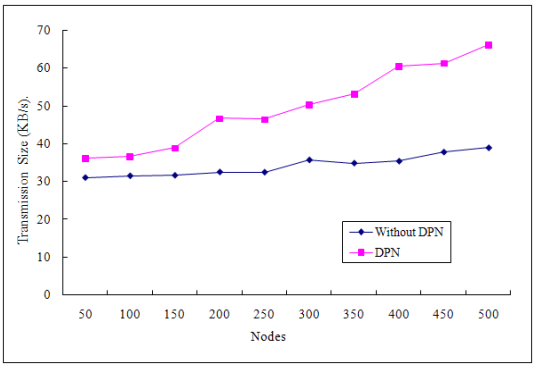


Figure 14. Avg. Transmission Size Per Node

V. Conclusion

A method to approximate physical neighbor discovery for P2P-VEs is proposed in this paper. Peers with short latencies are grouped by binning first, and then are maintained via a tree structure. We show that requesting content from physically close neighbors can much reduce the request latency while the system remains scalable (e.g., latency can improve by as much as 45%). The preservation of groups with a tree-structure is non-trivial if the number of peers is large, especially when many peers leave the system. As the dynamic joining and leaving of peers (i.e., churn) has not been considered, we hope to address these issues as future work.

References

[1] S. Singhal and M. Zyda, Neworked Virtual Environments: Design and Implementation, New York: ACM Press, 1999.

[2] http://vast.sourceforge.net/relatedwork.php

[3] S. Y. Hu, S. C. Chang, J. R. Jiang, "Voronoi State Management for Peer-to-Peer Massively Multiplayer Online Games," in *Proc. NIME*, 2008.

[4] S. Y. Hu, J. R. Jiang, and B. Y. Chen, "Peer-to-Peer 3D Streaming," *IEEE Internet Computing*, to appear.

[5] E. Teler and D. Lischinski, "Streaming of complex 3d scenes for remote walkthroughs," *CGF (EG 2001)*, vol. 20, no. 3, 2001.

[6] N. Magharei and R. Rejaie, "Prime: Peer-to-peer receiver-driven meshbased streaming," in *Proc. INFOCOM*, 2007, pp. 1415–1423.

[7] S. Y. Hu et al., "FLoD: A Framework for Peer-to-Peer 3D Streaming," in *Proc. INFOCOM*, Apr. 2008.

[8] 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.

[9] J. Chim et al., "Cyberwalk: A web based distributed virtual walkthrough environment," in *IEEE TMM*, vol. 5, no. 4, pp. 503–515, 2003.

[10] H. Miura and M. Yamamoto, "Content Routing with Network Support Using Passive Measurement in Content Distribution Networks", in *Proc. CCNC*, October 2002.

[11] M. Y. Hsieh, H. C. Yang, and L. M. Tseng, "Finding Nearest Neighbors in Replication-Aware CDN-P2P Architecture", *Journal of Internet Technology*, Vol.6, No.2, 2005

[12] S. Ratnasamy et al., "Topologically-aware overlay construction and server selection", in *Proc. INFOCOM*, 2002, pp. 1190- 1199.