Scalable AOI-cast for Peer-to-Peer Networked Virtual Environments

Jehn-Ruey Jiang, Yu-Li Huang, and Shun-Yun Hu
Department of Computer Science and Information Engineering
National Central University, Taiwan

Abstract

Networked virtual environments (NVEs) are computer-generated virtual worlds where users interact by exchanging messages via network connections. Each NVE user often pays attention to only a limited visibility sphere called area of interest (AOI) where interactions occur. The dissemination of messages to other users within the AOI (i.e., the AOI neighbors) thus is a fundamental NVE operation referred to as ***AOI-cast****. Existing studies on NVE scalability have focused on* ***system scalability****, or the ability for the system to handle a growing number of total users, by using multicast or peer-to-peer (P2P) architectures. However, another overlooked, yet important form of scalability relates to the handling of a growing number of users within the AOI (or* ***AOI scalability****). In this paper, we propose two AOI-cast schemes, called* ***VoroCast*** *and* ***FiboCast****, to improve the AOI scalability of P2P-based NVEs. VoroCast constructs a spanning tree across all AOI neighbors based on Voronoi diagrams, while FiboCast dynamically adjusts the messaging range by a Fibonacci sequence, so that AOI neighbors would receive updates at frequencies based on their hop counts from the message originator. Simulations show that the two schemes provide better AOI scalability than existing approaches.*

1. Introduction

Networked virtual environments (NVEs) are distributed systems where geographically dispersed users assume virtual representations called *avatars* to interact by exchanging network messages. Each user in a NVE can be seen as a coordinate point on a 2D plane with a bounded visibility called *area of interest* (AOI) [24]. AOI is often a circular area centered at the user, and other users within this area are called the *AOI neighbors*. As users need to be aware of the current states of its AOI neighbors, each user has to send messages about its state changes, such as its current position and actions, to users whose AOIs cover itself. If we assume that all AOIs are of the same size, as most NVEs do, then each user has to send messages to all of its AOI neighbors. In this paper, we use the term *AOI-cast* to refer such message propagation. As AOI-cast is a fundamental NVE operation, our goal is to develop efficient and scalable AOI-cast.

Creating scalable NVEs is an important topic, as successful NVEs often host many users. For example, Massively Multiplayer Online Games (MMOGs) are NVEs that support hundreds of thousands of simultaneous users. However, two forms of scalability exist for NVE systems: *system scalability* indicates a NVE's ability to handle a growing number of *total* users in the system; whereas *AOI scalability* refers to the ability to handle a growing number of users within a particular AOI. Both are important, but different properties of a NVE system.

Client-server is the main architecture for today's NVEs. However, since a server (or server cluster) has only limited resources at any given moment, server-only designs have inherently limited system scalability. Recent researches thus point to peer-to-peer (P2P) architectures to improve NVE's system scalability (e.g., Solipsis [15, 9], SimMUD [16], VON [11], N-tree [10], COVER [19], OPeN [8], APOLO [17], Skip Delaunay Network (SDN) [21], Colyesus [4], Peers@Play [23] and HyperVerse [5]). By distributing loads to all user nodes (or *peers*), clients become not just resource consumers but also providers. Besides system scalability, other P2P-NVE issues such as overlay partitioning [14], voice-chatting [13], secure messaging [6] and 3D streaming [12], have also been investigated.

While system scalability is studied extensively with P2P approaches, the issue of AOI scalability has received less attention. The problem is also known as the *hotspot*, *crowding* problem [11], or the user density challenge. AOI scalability is potentially improvable by a P2P-based AOI-cast. Existing schemes can be classified by how connections are made. On one end of the spectrum are *direct connection* schemes where each node exchanges messages with all AOI neighbors directly (e.g., Solipsis, VON, COVER, OPeN, and Colyseus). Direct connections allow small latency and robustness, but incur high bandwidth costs. When there are many AOI neighbors, peak bandwidth usage may exceed the peer's bandwidth limit (see Figure 1), causing overload

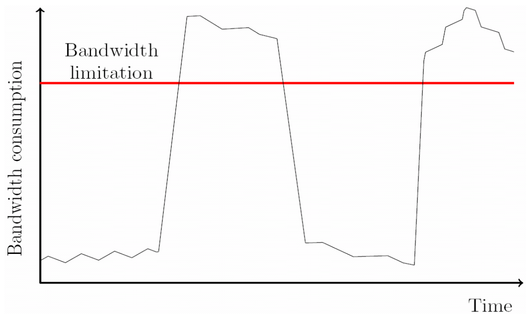

Figure 1. Bandwidth usage for AOI-cast

or degradation of the system. Towards the other end are *forwarding* schemes where the number of connections becomes less and messages are delivered by relaying. Forwarding schemes thus may achieve better AOI scalability at a cost of increased, but controllable latency, as the bandwidth load in a densely populated AOI is shared among the AOI neighbors. Forwarding schemes can be classified as forming structural trees (e.g., SimMud and N-tree) or spanning trees (e.g., APOLO, SDN, and VON-forwarding [7]). For structural trees (e.g., a quad-tree), each node only has a fixed number of connections. However, latency for message transmission may increase unevenly, as certain nodes could require more hops to reach. Bandwidth utilization also may not match each node's capability. Spanning tree designs thus are more flexible and will be our focus.

In this paper, we focus on forwarding AOI-cast due to its more scalable nature for both system and AOI scalability. We propose two forwarding AOI-cast designs, *VoroCast* and *FiboCast*, that improve the AOI scalability for P2P NVEs. VoroCast organizes AOI neighbors with a Voronoi diagram [3] where each peer only connects with the closest set of neighbors. Compared with other forwarding schemes, VoroCast provides non-redundant message disseminations in a bounded number of hops, and allows further bandwidth reduction with message packing and compression. FiboCast utilizes the fact that a user usually is more interested in changes that are nearer or clearer, and improves VoroCast by dynamically adjusting the message dissemination range according to a Fibonacci sequence, such that nodes with smaller hop counts from the sender receive messages more frequently than others. Consequently, FiboCast achieves better AOI scalability than VoroCast. We perform simulation experiments to evaluate the performances of both VoroCast and FiboCast, and compare them with related schemes.

The rest of the paper is organized as follows. We describe related work on P2P AOI-cast in section 2. In section 3, we present the design of VoroCast and FiboCast. In section 4, we evaluate our approaches by simulations and discuss the results. Conclusions are presented in section 5.

2. Related work

P2P AOI-cast can be seen as a form of *application-layer multicast* [18] within the scoped region of a user's AOI. It is thus similar to a *geocast* [20, 21] but performed in virtual space. Delaunay overlay [18] and CAN [22] are both general application-layer multicast over spatial domains. However, they do not specifically support scoped multicast or node movements. Subsequent work on Skip Delaunay Network (SDN) [21] improves by providing multicast within only a limited AOI. However, node positions are still assumed to be static and thus do not reflect NVE's dynamic nature. APOLO [17] is the first spanning tree AOI-cast that considers node movements. Each node connects with the closest neighbors in each of the four quadrants, and sends out messages to all AOI neighbors by forwarding. While the connection size in APOLO is relatively constrained, due to the limited connections, some routing paths become unnecessarily elongated, increasing the latency (see Figure 2, a message from node s4 to s8 is forwarded with additional hops). VON-forwarding [7] supports node movements and has potentially fewer routing hops than APOLO. As our work is based on VON and VON-forwarding, we first describe their designs as follows.

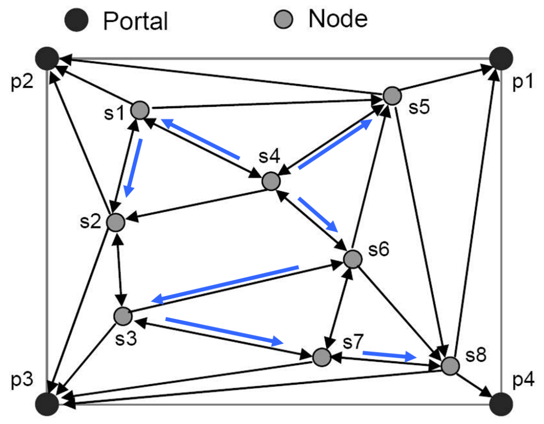

Figure 2. Elongated path in APOLO [17]

Each node in VON organizes its AOI neighbors with a Voronoi diagram [3] and separates them into various Voronoi regions. Figure 3 shows such an organization: for a center node u, if the big circle is its AOI, then the square nodes are its *enclosing neighbors* (i.e., nodes whose regions directly surround the Voronoi region of u); triangle nodes are its *boundary neighbors* (i.e., nodes whose regions overlap with its AOI boundary); stars are both enclosing and boundary neighbors; inverted triangle is a regular AOI neighbors; and diamonds are other invisible or irrelevant nodes.

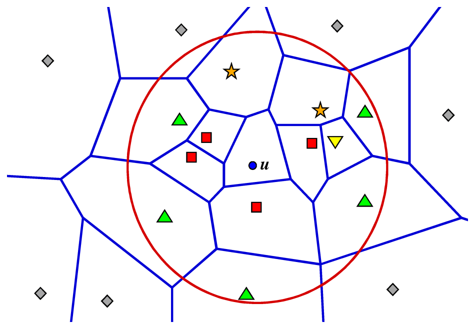

Figure 3. Different neighbor types in VON

In VON, a node directly connects to all of its AOI neighbors to send its current position and states periodically. The boundary neighbors should also preform *neighbor discovery* (i.e., the notification of new AOI neighbors) when they receive position updates from a moving node, so that a moving node may learn of new AOI neighbors. This is possible because new AOI neighbors are always the enclosing neighbors of existing boundary neighbors. VON thus has low message overhead, short latency and high consistency for node positions [11].

VON-forwarding [7] is based on VON, but each node connects only to the enclosing neighbors rather than all AOI neighbors. Since a node's enclosing neighbors are limited (e.g., 6 on average [18]), each node's connection size is basically constant. When sending messages, the message is first sent to each of the enclosing neighbors, which in turn forwards the message to their own enclosing neighbors. The forwarding continues until all AOI neighbors have the message. VON-forwarding also uses message packing and compression to reduce the overall bandwidth usage. However, redundant messages may be sent during the forwarding, which incurs unnecessary overhead (see Figure 4).

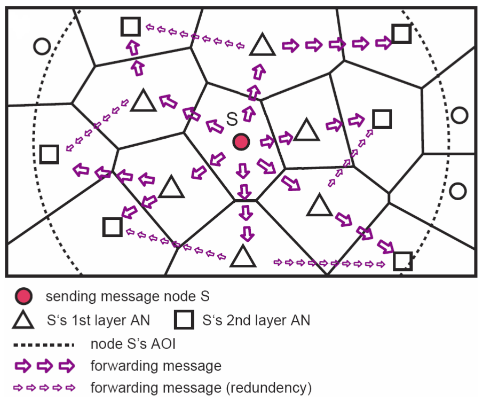

Figure 4. VON forwarding model [7]

3. Scalable AOI-cast

3.1. VoroCast

The basic idea of VoroCast is to construct a multicast tree spanning all AOI neighbors for each node. Messages can then be sent along the tree edges without redundancy. As in VON, the AOI is partitioned by a Voronoi diagram based on AOI neighbors' coordinates. Each node has a unique ID and two types of neighbors. The first is called *one-hop neighbors*, which are basically the *enclosing neighbors* in VON. Note that our definition differs from the *1-hop neighbors* in APOLO [17], which is a superset of the enclosing neighbors. The second is called *two-hop neighbors*, which are the one-hop neighbors of one-hop neighbors except for redundant ones. Nodes always connect to their one-hop neighbors to exchange the *one-hop neighbor lists* (containing the neighbors' IDs, positions and IP addresses) so that two-hop neighbors are properly known. Messages generated by a root node r are first sent to all of r's one-hop neighbors. Upon receiving the message, an intermediate node x then forwards it to uniquely selected child nodes within r's AOI.

To construct the spanning tree, no two nodes should select the same node as child to avoid transmission redundancy. We observe that while many potential child nodes exist when growing a tree from a root, it is possible to select just one parent for every node, given a root location (e.g., the closest to the root may be selected as parent). Therefore, to uniquely select a child, we can do so from the perspective of the child that is selecting a parent.

In the following child selection procedure (Figure 5), $x.N$ stands for the one-hop neighbor set of node x; $x.P$ is the parent of x; $r.AN$ is the AOI neighbor set of r; and $min\_dist(S, r)$ refers to the node from the node set S with minimal Euclidean distance to r. To begin, node x checks for each one-hop neighbor y, excluding its own parent or any direct children of the root (i.e., $y \in ((x.N - x.P - r.N) \cap r.AN)$). If x is the closet to r among all of y's one-hop neighbors, it would be y's parent. Node y is thus selected as one of x's children to receive forwarded messages. Upon receiving a message, y would also record x as its parent. Note that when two nodes are equidistant from r, their unique IDs can break the tie for choosing the parent.

```
// node $x$ selects a child node to forward $r$'s messages
SelectChild($x, r$)
// find out which child $y$ would select $x$ as its parent
for each $y \in ((x.N - x.P - r.N) \cap r.AN)$
    $z = min\_dist(y.N, r)$;
    if ($z$ equals $x$)
        add $y$ as a child of $x$;
```

Figure 5. Child selection procedure

After the selection, a node forwards messages to all of its children. Note that when several sources are sending concurrently, multiple messages may need to be delivered to the same one-hop neighbor. In such case, the messages can be packed into a single packet to save headers, and be compressed with a higher compression ratio than the unpacked messages. Message packing and compression thus may reduce bandwidth usage significantly.

3.2. FiboCast

VoroCast utilizes the bandwidth of neighboring nodes to relieve a sender from using up its own bandwidth. However, when nearby nodes are sending simultaneously (e.g., when they move at the same time), each node has to send its own messages plus those for the neighbors. Consequently, bandwidth depletion may still occur. We note that users in NVEs may pay more attention to activities that are more obvious in the vicinity. For example, the left side in Figure 6 is more crowded. So although both node u and node v are roughly equidistant from the center node, as node u is in the less crowded area, it may observe the center node more easily. On the other hand, the center node may appear obscured to node v due to the other users in between. In such a case, it makes sense for the center node to update u more frequently than v, and we can adaptively adjust the transmission frequency so that neighbors with more hop counts away receive messages less frequently. In this way, bandwidth usage can be reduced while keeping reasonable interactivity. Note that such reduction is applicable only to message types that could tolerate occasional losses (e.g., movement or voice packets) but not those that need to be reliable (e.g., chat or trade messages).

FiboCast adjusts the message frequency based on a Fibonacci sequence, which contains a series of numbers where each is the sum of the two previous ones (e.g., 0, 1, 1, 2, 3, 5, 8...). Different sequences can thus be created with different initial numbers. We add two more fields to each message in FiboCast: a *current hop count* and a *maximal hop count*, where the latter is set by the Fibonacci sequence in a round-robin fashion. Each message forwarding would increment the *current hop count* (i.e., similar to a time-to-live), and forwarding stops when the *maximal hop count* is reached. As Fibonacci grows slowly at first but quickly later, with such a series of maximal hop counts, nodes would receive messages with gradually decreasing frequency if they are more hops away from the root. Note that *infinity* is attached to each sequence so that all AOI neighbors can eventually receive messages. A minimal starting number of 2 is also assumed to allow proper two-hop node discovery and child selection (e.g., for a sequence of $<0, 1, 1, 2, 3, 5, 8, \infty>$, the maximal hop counts would be 2, 3, 3, 4, 5, 7, 10, ∞, 2, 3, 3, 4, 5, 7, and so on).

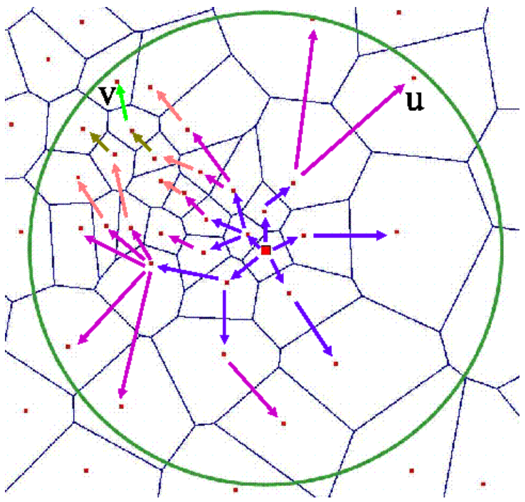

Figure 6. An unevenly crowded situation

4. Evaluation

In this section, we evaluate VoroCast and FiboCast by simulations in terms of their bandwidth consumption, neighborship consistency and drift distance. We also compare them against the original VON scheme (referred simply as VON from now on). Since VoroCast and FiboCast do not have message redundancy and will always incur less messages than VON-forwarding, we do not include VON-forwarding in the following comparisons. We first introduce the simulation environment, and then describe the simulation results.

4.1. Simulation environment

We base our simulations on the open source *VAST* [1] library. In our discrete-time simulations, we assume that nodes are initially placed at random positions on a 2D plane that is 1000x1000 units. Each node has a fixed AOI radius of 200 units and revokes VoroCast or FiboCast to send position updates in every discrete *time step*. A node moves according to random waypoint pattern [11] at a constant speed of 5 units per step, and each simulation lasts for 1000 steps. For simplicity, we assume that there is no packet loss and the transmission latency is constant, where messages are received and processed in the next time step after their delivery. We also assume that nodes have unlimited bandwidth, so that we can see how much bandwidth use is incurred for a given simulation. For both VoroCast and FiboCast, message compression is also enabled with the *zlib* compression library [2].

4.2. Simulation results

4.2.1 Bandwidth consumption

Bandwidth usage is a good indicator for a system's scalability, as the main resource bottleneck in large-scale NVEs is the bandwidth [11]. Systems are scalable as long as the bandwidth usage of each node remains bounded. Figure 7 shows the per-node, per-second bandwidth consumption of VoroCast, FiboCast and VON. We can see that the proposed schemes incur less bandwidth than VON. This is because our schemes are based on forwarding and can apply message packing and compression to reduce bandwidth usage. If we set a bandwidth limitation for each node, we can see that FiboCast accommodates the most nodes, followed by VoroCast and VON (e.g., if each node has only 15 KB/s bandwidth, then FiboCast, VoroCast and VON can accommodate a total of around 1000, 500, and 200 nodes, respectively). Note that by simulating a growing number of nodes with a fixed AOI, our simulations test the performance for both system and AOI scalability (i.e., AOI scalability is directly related to system scalability in our scenarios).

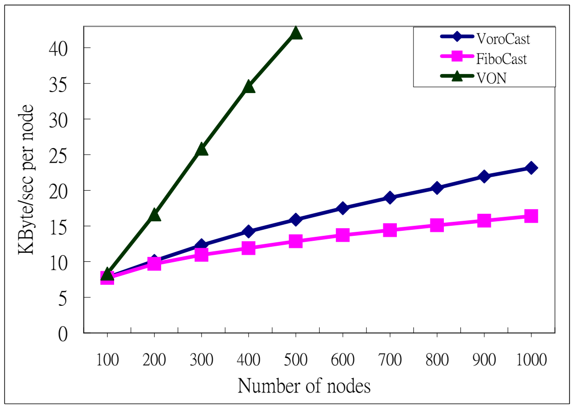

Figure 7. Bandwidth consumption

4.2.2 Neighborship consistency

While the scalability of VoroCast and FiboCast may be superior than the original VON, we should also evaluate their main drawbacks in terms of the reduced consistency due to more latency and less updates. Neighborship consistency [14] can be a good indicator and is defined as follows.

$$NC_i = \frac{KN_i}{ANi}$$

NC_i is the neighborship consistency for node i, and is defined as the ratio of the number of known AOI neighbors KN_i to the number of total, actual AOI neighbors, AN_i. For example, if a node has 100 AOI neighbors but is only aware of 90, its neighborship consistency is 90%. A global neighborship consistency NC can also be calculated by averaging those from the individual nodes. Figure 8 shows the global neighborship consistency of VoroCast, FiboCast and VON. In VON, each node sends messages to all AOI neighbors directly, so it maintains very high consistency. The consistency of VoroCast is above 95%, which is acceptable. The overall consistency of FiboCast is worse than VoroCast. However, if we look more closely at the consistency within a smaller zone from the AOI center (e.g., 25%, 50%, 75% and 100% of the AOI radius from the center), we see that neighborship consistency is actually higher than 95% for the 25%, 50%, and 75% zones, which means that for practical purposes good consistency is still maintained.

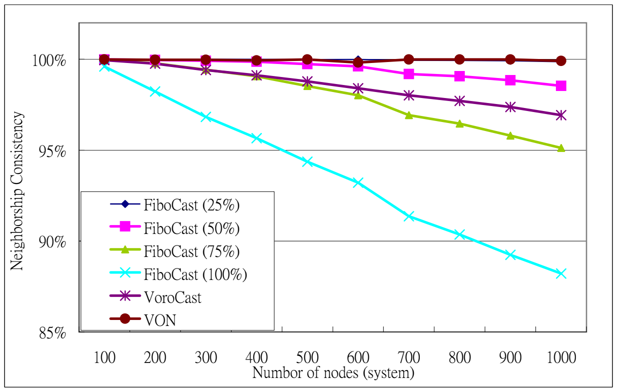

Figure 8. Neighborship consistency

4.2.3 Drift distance

Drift distance is another way to measure the consistency in node positions, and is the difference between the real and observed positions of nodes. It is defined as follows [11].

$$DD_i \quad = \quad \underset{j \in AOI_i}{\text{AVG}} \; |OP_j - RP_j|$$

DD_i is the average drift distance for node i calculated from the set of i's known AOI neighbors (AOI_i). OP_j and RP_j are, respectively, the observed and the actual positions of a known AOI neighbor j of node i. Again, a global drift distance DD can be calculated by averaging those from the individual nodes. Drift distance reflects the accuracy of AOI neighbor positions, and is basically determined by message latency. Since FiboCast adjusts the update frequency adaptively, its latency is highly variable, and we exclude it from the comparisons.

Figure 9 shows the drift distances of VoroCast and VON for 50 to 500 nodes. Drift distances for VON are near 0 since all updates are transmitted in one hop. VoroCast trades drift distance for AOI scalability: its drift distances grow gradually with the number of AOI neighbors. We also observe that the drift distances grow proportionally to the number of hops in the communication path. For example, the drift distance of VoroCast is about 12.8 units for 500 nodes, when the average path is 7.5 hops; and around 8.7 units for 300 nodes, when the average path is about 5.2 hops. However, these drift distances are acceptable when compared to an AOI radius of 200 units.

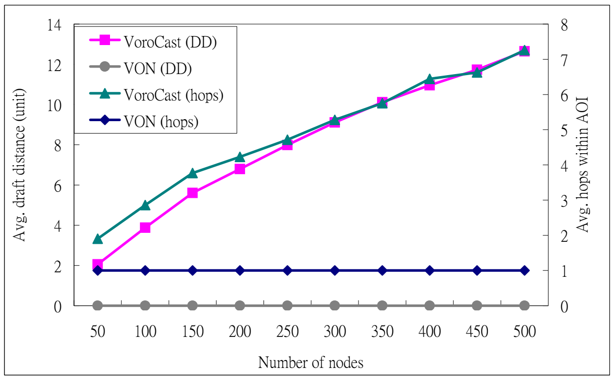

Figure 9. Drift distance and latency

5. Conclusion

In this paper, we propose two AOI-cast schemes for P2P-based NVEs: VoroCast and FiboCast. They support better AOI scalability than existing direct connection approaches. VoroCast is based on Voronoi diagrams to construct a tree spanning all AOI neighbors of a message originator, so that a sender's peak bandwidth usage is reduced by distribution to neighbors. Further bandwidth reduction can also be achieved via message packing and compression. FiboCast is useful for loss-tolerable frequent updates such as position or voice packets. It dynamically adjusts the message dissemination range according to a Fibonacci sequence so that neighbors with fewer hop counts from the AOI center can get messages more frequently than those further away. Our simulation results show that VoroCast and FiboCast consume less bandwidth than the direct connection scheme VON, thus achieving better AOI scalability, while keeping relatively fair neighborship consistency and drift distances. One potential drawback of our schemes is that the child node degrees are based on peers' positions but not bandwidth capacities, which may be a problem in some cases. Also, for FiboCast, how to select a proper segment from the Fibonacci sequence for specific environments remains unsolved. Addressing these issues will be our future work.

References

[1] VAST Project. http://vast.sourceforge.net/.

[2] zlib Project. http://www.zlib.net/.

[3] F. Aurenhammer. Voronoi diagrams - a survey of a fundamental geometric data structure. *ACM CSUR*, 23(3):345–405, 1991.

[4] A. Bharambe, J. Pang, and S. Seshan. Colyseus: A distributed architecture for multiplayer games. In *Proc. ACM/USENIX NSDI*, 2006.

[5] J. Botev et al. The hyperverse - concepts for a federated and torrent-based "3d web". In *Proc. MMVE*, 2008.

[6] M.-C. Chan, S.-Y. Hu, and J.-R. Jiang. A Efficient and Secure Event Signiture (EASES) Protocol for Peer-to-peer Massively Multiplayer Online Games. *LNCS*, 4476, 2007.

[7] J.-F. Chen et al. A Forwarding Model for Voronoi-based Overlay Network. In *Proc. P2P-NVE*, 2007.

[8] S. Douglas et al. Enabling massively multi-player online gaming applications on a p2p architecture. In *Proc. ICIA*, pages 7–12, 2005.

[9] D. Frey et al. Solipsis: A decentralized architecture for virtual environments. In *Proc. MMVE*, 2008.

[10] C. GauthierDickey, V. Lo, and D. Zappala. Using n-trees for scalable event ordering in peer-to-peer games. In *Proc. NOSSDAV*, pages 87–92, 2005.

[11] S. Hu, J. Chen, and T. Chen. VON: a scalable peer-to-peer network for virtual environments. *IEEE Network*, 20(4):22–31, 2006.

[12] S.-Y. Hu et al. FLoD: A Framework for Peer-to-Peer 3D Streaming. In *Proc. IEEE INFOCOM*, 2008.

[13] J.-R. Jiang and H.-S. Chen. Peer-to-Peer AOI Voice Chatting for Massively Multiplayer Online Games. In *Proc. P2P-NVE*, 2007.

[14] J.-R. Jiang, J.-S. Chiou, and S.-Y. Hu. Enhancing Neighborship Consistency for Peer-to-Peer Distributed Virtual Environments. In *Proc. ICDCS workshops (CDS 2007)*, 2007.

[15] J. Keller and G. Simon. Solipsis: A massively multi-participant virtual world. In *Proc. PDPTA*, 2003.

[16] B. Knutsson et al. Peer-to-peer support for massively multi-player games. In *Proc. IEEE INFOCOM*, 2004.

[17] J. Lee et al. APOLO: Ad-hoc Peer-to-Peer Overlay Network for Massively Multi-player Online Games. Technical report, KAIST, 2006.

[18] J. Liebeherr and M. Nahas. Application-layer multicast with delaunay triangulations. In *Proc. IEEE INFOCOM*, pages 1651–1655, 2001.

[19] P. Morillo, W. Moncho, J. Orduna, and J. Duato. Providing full awareness to distributed virtual environments based on peer-to-peer architectures. *LNCS*, 4035:336–347, 2006.

[20] J. Navas and T. Imielinski. Geocast addressing and routing. In *Proc. ACM/IEEE MOBICOM*, 1997.

[21] R. Nishide et al. Data aggregation method for view range computation on p2p-based vcs. In *Proc. MMVE*, 2008.

[22] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Application-level multicast using content-addressable networks. In *Proc. NGC*, 2001.

[23] G. Schiele et al. Requirements of peer-to-peer-based massively multiplayer online gaming. In *Proc. GPC*, 2007.

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