Scalable Peer-to-Peer Networked Virtual Environment

Shun-Yun Hu
Dept. of Computer Science and Information Engineering
Tamkang University
Tamsui, Taipei County 251, Taiwan
syhu.tw@yahoo.com.tw

Guan-Ming Liao
Institute of Physics
Academia Sinica
Taipei 11529, Taiwan
gm.liao@msa.hinet.net

ABSTRACT
We propose a fully-distributed peer-to-peer architecture to solve the scalability problem of Networked Virtual Environment in a simple and efficient manner. Our method exploits locality of user interest inherent to such systems and is based on the mathematical construct *Voronoi diagram*. Scalable, responsive, fault-tolerant NVE can thus be constructed and deployed in an affordable way.

Categories and Subject Descriptors
C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design --- Distributed networks

General Terms: Algorithms, Design

Keywords
Networked Virtual Environment (NVE), peer-to-peer (P2P), massively multiplayer (MMP), Voronoi diagram, scalability, interest management

1. INTRODUCTION
Networked Virtual Environments (NVEs) [13] are computer-generated, synthetic worlds that allow simultaneous interactions of multiple participants. Since the early days of SIMNET, a U.S. Government project for large scale combat simulations, to the recent boom of Massively Multiplayer (MMP) Online Games (MMOG) [8], efforts to allow people to interact in realistic, immersive virtual environments have gone a long way. Science fiction works, such as Neal Stephenson's novel *Snow Crash* and the recent *Matrix* movie trilogy, serve as inspirations to many for the eventual creation of a 3D environment that is truly massive, persistent, realistic and immersive. With rapid technology developments, converging advances in CPU, 3D accelerator, and bandwidth may make the vision come true in the foreseeable future. However, a number of issues exist in the creation of a large-scale NVE, namely:

Consistency - For meaningful interactions to happen, each user's experiences in the virtual world must be more or less consistent. This includes maintaining states and keeping events synchronized.

Performance / Responsiveness - NVEs are simulations of the real world. Responsiveness therefore is important for immersion. However, performance requirements vary between applications.

SIGCOMM'04 Workshops, Aug. 30 & Sept. 3, 2004, Portland, OR, USA.

Security - Most NVEs allow people to engage competitively (e.g. combat or treasure hunt). User authentication and fairness against cheating therefore are required. In fact, this is often the most concerned issue by commercial NVE developers.

Scalability – Scalability usually concerns with the number of simultaneous users in NVE [13]. It is important in two respects: (1) Content possibility. Certain game plays are only realizable when many people participate, such as community and social oriented game play. (2) Service availability. Large-scale NVEs are similar to websites, where usage may increase dramatically and unexpectedly. Systems will break if they are not scalable.

Persistency - To create sophisticated contents, certain data, such as user profile and valuable virtual objects, must be persistently stored and accessed between user sessions.

Reliability / Fault-tolerance – User experience is negatively affected if a play session suddenly breaks down due to server failure. Reliability is thus important to make NVE a service with quality.

The first three issues exist for all multiplayer virtual environments. Remaining ones are additional criteria for MMP applications. We consider scalability the most important issue if we plan to build truly massive worlds and applications, which millions of people can enjoy. Current approaches to scalability mostly include setting up multiple servers or server-clusters. However, maintaining server resources is costly and has inherent design limitation, as we will discuss in the next section.

This paper proposes a fully-distributed peer-to-peer (P2P) architecture, which attempts to solve the scalability problem based on the mathematical construct *Voronoi diagram* [4]. The main contribution of the paper is the proposal of a very simple and resource-efficient solution to the difficult scalability problem. Our solution dramatically reduces server load and can be achieved with a single lightweight server. In fact, the server only serves as an access point that provides authentication, and is not required after login. This will allow scalable NVEs to be built affordably.

In order to simplify the problem, two assumptions are made: (1) user computers can be trusted (2) message exchange itself is sufficient for maintaining the states of the world (i.e. user position, user actions, and temporary objects). In other words, currently we do not take security and persistency issues into consideration.

We will analyze the scalability problem in Section 2, and will present our design in Section 3. Our current implementation is described in Section 4. Finally, Section 5 concludes the paper.

2. THE SCALABILITY PROBLEM

2.1 Theoretical Analysis
Scalability is a phenomenon observed in many natural and artificial systems. We see systems that accommodate components (or nodes) in a wide range of numbers as being "scalable".

There are two main characteristics for scalable systems:

- **Joinability:** components or nodes may be added to the system.
- **Maintainability**: system remains functional after various nodes enter or leave the system.

Existing resources in any given system is usually finite, and are consumed at end-point when a new node is added. For example, bandwidth of existing routers is consumed when adding new routers. A system is "joinable" only when the accepting node has enough spare resources. Likewise, maintainability is sustained only when resource is not depleted after the new node joins. Two more properties exist to counter resource depletion:

- **Resource-growing**: useful system resources (i.e. resource at the accepting nodes) increase with the addition of new nodes
- **Decentralized end-point resource consumption**: addition of a node does not consume some "centralized" resource.

Resource-growing is a general strategy found in almost all scalable systems (reducing consumption works to the same effect). In the Internet, although additional router consumes bandwidth, it also contributes new resource to accommodate new routers. Decentralized resource consumption, on the other hand, is not necessarily required. As long as resources are available at the accepting node, system can be "joinable" and "maintainable" even if it is done in a centralized fashion. For example, in a server-cluster, as long as server resources (bandwidth and processing capability) can be increased, then scalability is maintainable [1]. However, most massively scalable systems (such as Internet) exhibit decentralized resource consumption.

From the above discussion, we expect that to build a truly scalable NVE, one that may accommodate more users by orders of magnitude than existing systems, we need architectures that can grow its resource, and does not require centralized resource when additional users join.

2.2 Previous Work

Scalability for NVE generally concerns with whether the system can accommodate a large number of simultaneous users. Various approaches have been taken, and generally fall into either the "increase resource" or the "reduce consumption" categories:

Increase Resource. Using multiple servers for multiple worlds or using server-cluster to maintain a single world has become a popular approach, especially for commercial NVEs [1] [14]. For example, commercial MMOGs are set up with multiple servers for the same game, each serving a pre-determined maximum number of users. When a server is full, it simply denies additional connections. Total number of players can be very large. (For example, a record of 160,000 concurrent users was reported for *Lineage* in 2002 in Taiwan.) However, users between different servers may not interact, and some systems do not even share user profiles. Server-cluster [3], on the other hand, divides the virtual world into *regions* or *zones*, and supports what appears to users as a single coherent world. Since server-cluster offers desirable properties, it has become the trend for building large-scale NVEs. However, server-centered approach is costly for server-side bandwidth, hardware, and maintenance, which limits the number of potential NVE developers.

Decrease Consumption. The central theme to this approach is *interest management* [12]. While other techniques to economize bandwidth exist, such as packet compression or aggregation [13], we consider interest management more relevant. Messages are generated by user actions and exchanged to maintain consistency [13]. However, if messages are sent to all other users, the amount of transmission and processing grows at $O(n^2)$, which is clearly not scalable. Real-world observation tells us that each individual only has a limited visibility or "sphere of interaction". In other words, our interest is localized [12]. Interest management therefore deals with relevant information filtering, to decrease unnecessary resource consumption while maintaining adequate interactivity. Early NVEs did not have interest management, and were set up by hosts broadcasting messages in the same LAN [13]. To provide interest management, later systems adopt the client-server model, where clients send messages to the server, which acts as interest manager and sends back filtered messages. Interest management can be based on various criteria. It can be distance-based (by geography), class-based (by object or user attributes), or some combination of both [12]. A commonly used concept is *Area of Interest* (AOI), which describes a circular or rectangular box centered on the user. Only messages generated within AOI are relevant to the user (Figure 1).

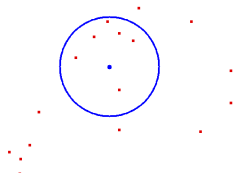

Figure 1: Each dot represents a user in the virtual world, and circle represents Area of Interest (AOI) of a particular user.

A common technique in interest management is to divide the world into various *regions*. Each user only receives messages (position update or interaction message) from relevant regions. This can be done by server-side message filtering, or via network-support such as multicast [10]. However, region size can be difficult to determine (Figure 2). If it is larger than AOI, irrelevant messages are still received; while if it is smaller than AOI, it becomes inefficient to maintain (e.g. subscribing to too many multicast address). Ideally, regions would dynamically adjust size and shape based on current user location. The real challenge then is to create *individualized* region that moves with the user.

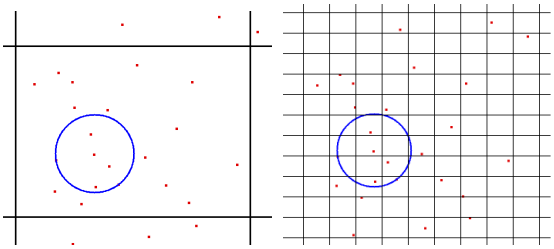

Figure 2: Difficulty in choosing region size. (L) AOI is smaller than region. (R) AOI is larger than region.

2.3 Promise and Challenge of P2P NVE

We consider the main challenge in scalability as: how to construct "resource-growing" and "decentralized consumption" into an architecture that also exhibits ideal interest management?

Peer-to-peer architecture naturally comes to mind, and appears to be an attractive alternative to client-server. Each participant contributes resource to maintain the system without consuming any

centralized resource. It matches with our criteria nicely. P2P architecture can be made very efficient if we only connect to relevant users (i.e. those within the AOI). (Keller and Simon describe this property as keeping *Local Awareness* [6].) A number of P2P overlay networks have been proposed in recent years: CAN, Chord, Pastry [8], and Hypercast [9], to name a few. However, these overlay networks mainly deal with setting up a distributed hash table (DHT) that maps keys to values and allows for content-lookup and retrieval (as in distributed file-sharing). While it is possible to build NVE on such overlay, as recently proposed by Knutsson *et al*. [8], there is overhead associated with using the overlay.

Common questions to all P2P networks are: correct topology maintenance and efficient content retrieval. Since a single node does not have knowledge of global topology or content location [7], these become difficult questions important to any P2P design. For topology maintenance, there are two issues to consider: whether it is *fully-connected* (described by Keller and Simon as the *Global Connectivity* property [6]) and whether all nodes have a *consistent view* of the topology. Unlike file-sharing P2P, where the desired content changes with user preference randomly and unpredictably, for P2P NVE the desired content is easier to identify -- messages generated by other users within the AOI. If only such messages are received, then message flow is managed optimally. So the "content discovery problem" for P2P network translates naturally to a *neighbor discovery problem* in P2P NVE.

Neighbor discovery is challenging, because there is a paradox in maintaining consistency in decentralized systems, as described by Makbily *et al*. [11]. At least three recently published papers offer different solutions: (1) Knutsson *et al*. describe P2P support for Massively Multiplayer Games by using Pastry and Scribe, a P2P overlay and its associated simulated multicast [8]. The virtual world is divided into regions of fixed-size. Each region is managed by a promoted node called *coordinator*, which serves as the root of a multicast tree. Users inside the same region subscribe to the address of the root node to receive updates from other users, so neighbors are discovered via the coordinator. Coordinators maintain links with each other, facilitating user transition to other regions. (2) Kawahara *et al*. describes a fully-distributed scheme where each user keeps track of a fixed number of nearest neighbors [5]. Nodes constantly exchange neighbor list with their own neighbors. After sorting through the list by distance, each node may learn of new nodes and update existing links. (3) Solipsis [6] is also a fully-distributed system, where each node attempts to link with all the nodes within its AOI. Neighboring nodes serve as the "watchmen" for any approaching foreign nodes. Neighbor discovery is done by notification from known neighbors.

However, each of them incurs some undesirable properties. In Knutsson *et al*., since fixed region size does not reflect true AOI, users cannot see across regions. If users decide to listen to more regions, as suggested in the paper, unnecessary messages beyond AOI will be received. A more serious problem is the performance penalty incurred by using P2P overlay. As the overlay does not consider AOI, messages may need to be relayed by other nodes (1 to 2 hops for most cases, but in some cases it goes beyond 50. Note too that this is "virtual hop", so more delays happen at the physical level). In short, the architecture does not fully utilize the power of direct connections. In the Kawahara *et al*. approach, direct links are maintained between neighbors, so hop-count is most efficient (e.g. one virtual hop). However, constant exchange of neighbor list incurs network overhead (if 10 nearest neighbors are kept, one exchange requires receiving updates of 10x10 nodes). The more serious problem is keeping the topology connected. Since only a finite number of nearest neighbors are maintained, groups of users may lose contact to each other if separated by a large distance. The underlying overlay can thus separate into isolated parts [5]. Solipsis also uses direct links among neighbors (there is no relay). Additionally, it requires that each node be inside a convex hull formed by its neighbors in 2D plane. This way the topology is guaranteed to be fully connected (i.e. *Global Connectivity* is kept). However, inconsistent topology may happen during normal operation (though rare), since an incoming node may be unknown to directly connected neighbors, proper neighbor discovery is not guaranteed (i.e. *Local Awareness* is not kept, see Figure 3).

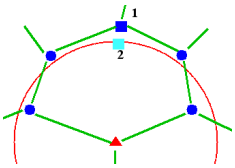

Figure 3: Undiscovered node in Solipsis. Lines are connections. Square node is not discovered as it moves from position 1 to 2. Topology is inconsistent though fully-connected.

3. VORONOI-BASED P2P NVE

3.1 Design

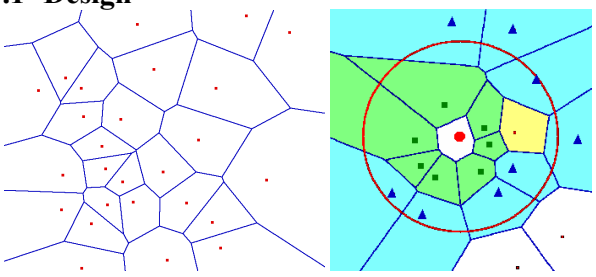

Figure 4: (L) Voronoi diagram. (R) Square (□): *enclosing neighbors*, triangle (Δ): *boundary neighbors.*

In this section, we will explain the design and analysis of our P2P approach, which is based on a well-studied mathematical construct *Voronoi diagram* [4]. Given *n* points on a plane (each point called a *site*), a Voronoi diagram is constructed by partitioning the plane into *n* non-overlapping *regions* that contain exactly one *site* in each *region*. A *region* contains all the points closest to the *region*'s *site* than to any other *site* (Figure 4L). The entire plane is therefore divided into arbitrary sizes in a deterministic way. Voronoi diagram can be used to find the *k*-nearest neighbors of a specific *site*. By using Voronoi, we may be able to identify *enclosing* and *boundary neighbors* for a given node. *Enclosing neighbors* are defined as *regions* that share a common edge with a given node's own *region*. *Boundary neighbors* are defined as *regions* that overlap with the node's AOI boundary (Figure 4R). Note that an *enclosing neighbor* can also be a *boundary neighbor*. These properties will help to solve the neighbor discovery problem described earlier.

The basic idea of our approach is to let each node construct and maintain a Voronoi diagram, based on the spatial coordinates of neighbors within the node's AOI. Each node keeps P2P connections with all neighbors that constitute the Voronoi. Connections are therefore based on spatial relationship in the NVE (not physical

network proximity). In our basic model, we assume that all AOIs are of the same radius, and are determined in an application-specific manner by the designer. Although a node only knows about a limited number of neighbors, it can learn of other new neighbors with the help of its *boundary neighbors*. Each peer serves as the "watchman" for one another in discovering approaching neighbors.

When the node moves, position updates are sent to *all* neighbors recorded in the Voronoi. If the receiver is a *boundary neighbor* (as determined by the sender), an overlap-check is performed. The receiver checks if the mover, with its new AOI, would enter into any of its *enclosing neighbors'* Voronoi region. The receiver only notifies the mover if a *new* overlap occurs (i.e. previously non-overlapped region becomes overlapped). This allows the moving node to get aware of potentially visible neighbors outside the AOI with minimal network overhead (only normal movement message is used). In case of a node leave or failure, its neighbors simply update their Voronoi after detection (through a loss of TCP connection or inactivity timeout). If the leaving user is considered a *boundary node*, queries are sent to discover any replacement (Figure 7).

3.2 Procedure

We will describe the basic procedures for joining, moving and leaving in the P2P NVE. The emphasis of these procedures is to maintain P2P topology consistency in a message-efficient manner.

Join

1. *Joining node* contacts the *gateway server* for a unique ID.
2. Join request is forwarded to *acceptor region* (defined as the region that contains the joiner's coordinates) via neighboring nodes with simple greedy forward (see Figure 5L).
3. *Acceptor node* sends back a complete list of its own neighbors.
4. *Joining node* contacts each neighbor on the list.
5. *Joining node* builds up a new Voronoi while other nodes update their Voronoi to accommodate the *joining node* (see Figure 5R).

Move

1. *Moving node* sends position coordinates to all neighbors (i.e. *boundary*, *enclosing*, and other neighbors). Messages for *boundary neighbors* are specifically marked.
2. *Boundary neighbor* will check if the *moving node*'s new AOI becomes overlapped with any of its *enclosing-neighbor's* Voronoi regions. If so then it sends a notification. (Figure 6L).
3. If a new neighbor is found, the *moving node* connects to it.
4. *Moving node* disconnects any *boundary neighbors* whose Voronoi region no longer overlaps with its AOI (Figure 6R).

Other actions (jump, chat, trade)

1. Send message to relevant neighbors recorded in the Voronoi. (For example, a private chat is directed only to neighbor(s) in the conversation, but an action such as "jump" is sent to all neighbors that can see the action, i.e., those in the Voronoi.)

Disconnect/Leave

1. *Leaving node* notifies with a list of its *enclosing neighbors.*
2. Neighboring nodes affected by the disconnection update their Voronoi. If the leaving node is seen as a *boundary neighbor*, then new *boundary neighbors* may be assigned.
3. For abnormal departure of *boundary neighbor*, a request for *enclosing neighbor* list is sent to known neighbors to ensure that topology remains consistent (see Figure 7).

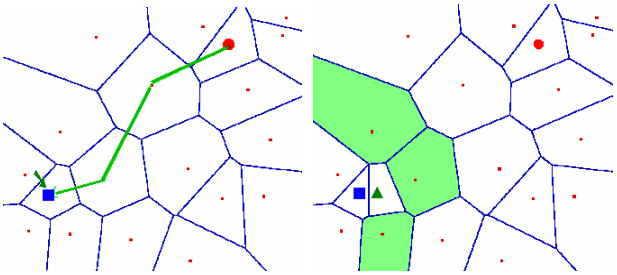

Figure 5: Join procedure. (L) Forward of join request. Circle is *gateway server*. Arrow indicates the *acceptor node.* (R) Triangle is the new node, shaded regions are neighbors affected by join. Note that the effect is localized.

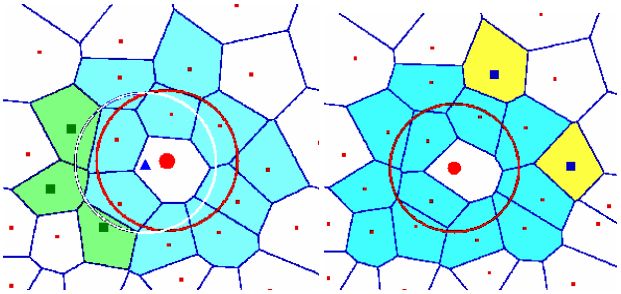

Figure 6: Move procedure. (L) Triangle indicates the intended new position. Squares are new neighbors about to be discovered. (R) After the move. Squares are the neighbors no longer overlap with AOI, therefore are disconnected.

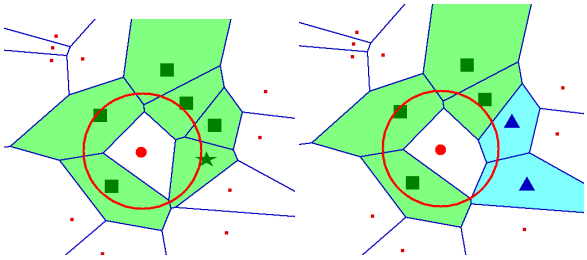

Figure 7: Leave procedure. (L) Before node leave, star is the leaving node. (R) After node leave, triangles (Δ) are the new boundary neighbors discovered with help of existing neighbors.

3.3 Analysis

A qualitative analysis of our current design is given below:

Consistency: In our design, as long as each node correctly keeps track of at least their *enclosing neighbors,* there is a guaranteed path between any two nodes; since discovery is covered in all directions, no node would be missed. P2P topology is therefore both *fully-connected* and *consistent* provided there is no network failure. Even if a small number of node fails, the network can still self-repair. This is an important improvement over existing approaches. Consistency in event synchronization is not currently guaranteed, because no "central authority" exists to decide the ordering.

Performance: Transmission hop-count between peers is optimally efficient as there is no relay. The low latency quality allows for responsive applications. When users are close to each other, *ideal interest management* is achieved (i.e. only messages within the AOI are received). However, if users are dispersed, connections with enclosing neighbor beyond AOI are needed to maintain the

topology (Figure 8L). Luckily, on average such neighbors are few, given Voronoi's characteristics (six on average [9], but *n*-1 neighbors in the worst-case, see Figure 8R).

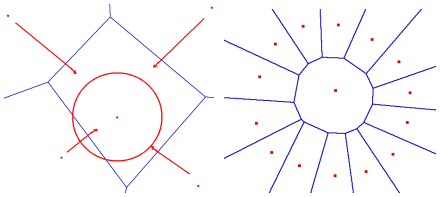

Figure 8: Potential issues with Voronoi. (L) Messages outside of AOI are still received to maintain P2P topology, but the amount is expected to be small. (R) Circular line-up of nodes.

There are two inherent disadvantages in the current design. (1) Each node must send duplicate messages to reach the neighbors, which requires more bandwidth than client-server (i.e. only one message is sent). (2) Since no message is processed centrally, aggregation or compression techniques cannot be leveraged.

Security: We assume all hosts can be trusted.

Scalability: The architecture matches the two criteria for scalable systems: resource-growing and decentralized consumption.

Persistency: We assume no persistent states in the current model.

Reliability: As long as each node maintains *some* reliable neighbors, the system should be able to self-repair inconsistency due to node failures. However, if a large number of nodes fail simultaneously, the P2P overlay may still be separated into mutually unaware parts. However, this is a general problem faced by all P2P networks that warrants future study.

4. IMPLEMENTATION

There are a number of existing algorithms for constructing and maintaining Voronoi diagrams [4]. The particular one we implement is Fortune's sweepline algorithm [2], which constructs Voronoi in O(n log n) time. Our design is currently implemented using High-Level Architecture Run-Time Infrastructure (HLA-RTI) interface, an IEEE standard originally proposed by the U.S. Department of Defense. We are investigating the suitability of the standard for MMOG applications. We name the current implementation Adaptive Scalable Cooperative Environment for Networked Dimensions (ASCEND). Our long-term goal is to develop an open source platform for NVE development.

5. CONCLUSION

We have presented a general picture of the scalability problem in NVE, and have analyzed requirements for potential solutions. A promising solution for the difficult neighbor discovery problem is also presented, by using Voronoi diagram. The general idea of our solution is to leverage knowledge of each peer about its neighbors to maintain the position states of all participants. Our solution is simple, efficient, and close to ideal interest management in NVE. Future works include variable-size AOI, persistency maintenance and security mechanism under P2P.

One of the most important concerns for commercial NVE is security, both for account information and game state authenticity (e.g. player's experience points, valuable virtual items). While accounts can be handled by a central server, user computers are always prone to hacking. This is the main reason why client-server is almost universally adopted. We feel that security might indeed be the main obstacle for commercial adoption of P2P, despite benefits in performance and cost. However, we believe that active research can be done to find "good enough" solutions. On the other hand, this "weakness" can be opportunity for other areas, such as education or social communities, where social interactions and collaborations are emphasized over competitions. One major feature of P2P NVE is its low cost, where scalability and performance is achieved affordably. For developers with limited budget, P2P NVE provides a promising alternative.

6. ACKNOWLEDGMENTS

We thank members of the WISE Lab in Tamkang Univ. and Joaquin Keller for valuable feedbacks, LSCP, Inst. of Physics, Academia Sinica for their facility, and Dr. Tzu-yang Chen for proofreading. We are very grateful of the helpful comments by the anonymous reviewers. We would also like to thank Prof. Wen-Bing Horng and Dr. Chin-Kun Hu for supporting this work. Special thanks to Prof. Jiung-yao Huang for introducing NVE to us.

7. REFERENCES

[1] Butterfly.net, Inc. The Butterfly Grid, 2003. www.butterfly.net/platform

[2] S. Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmic* 2, Pages 153-174. 1987.

[3] T. A. Funkhouser. RING: A Client-Server System for Multi-User Virtual Environments. In *Proc. of the 1995 Symposium on Interactive 3D Graphics*. pp. 85-92, Apr. 1995.

[4] L. Guibas, J. Stolfi. Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams. *ACM Trans. on Graphics*, Vol. 4, No. 2, Pages 74-123, April 1985.

[5] Y. Kawahara, T. Aoyama, H. Morikawa. A Peer-to-Peer Message Exchange Scheme for Large-Scale Networked Virtual Environments. *Telecommunication Systems* Vol. 25 Issue 3, Pages 353–370, 2004

[6] J. Keller, G. Simon. Solipsis: A Massively Multi-Participant Virtual World. In *Proc. of PDPTA 2003*. Pg. 262-268. 2003

[7] H. T. Kung, C. H. Wu. Hierarchical Peer-to-Peer Networks. Technical Report IIS-TR-02-015, Institute of Information Science, Academia Sinica, Apr., 2001.

[8] B. Knutsson, H. Lu, W. Xu, B. Hopkins. Peer-to-Peer Support for Massively Multiplayer Games. In *INFOCOM*, Mar. 2004.

[9] J. Liebeherr and M. Nahas. Application-layer Multicast with Delaunay Triangulations. In *Proc. of IEEE GLOBECOM*, Nov. 2001.

[10] M. R. Macedonia, M. J. Zyda, D. R. Pratt, D. P. Brutzman, P. T. Barham. Exploiting Reality with Multicast Groups. *IEEE Computer Graphics and Applications*, Volume 15, Issue 5, Pages: 38 - 45, Sept. 1995.

[11] Y. Makbily, C. Gotsman, R. Bar-Yehuda. Geometric algorithms for message filtering in decentralized virtual environments. In *Proceedings of the 1999 Symposium on Interactive 3D graphics*. Pages: 39 - 46. 1999.

[12] K.L. Morse. Interest Management in Large-Scale Distributed Simulations. Tech. Report ICS-TR-96-27, UC Irvine, 1996.

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

[14] Zona Inc. Terazona: Zona application frame work, 2003. www.zona.net/whitepaper/Zonawhitepaper.pdf.