Efficient Triangulation for P2P Networked Virtual Environments

Eliya Buyukkaya and Maha Abdallah
LIP6, University of Paris 6
{Eliya.Buyukkaya, Maha.Abdallah}@lip6.fr

ABSTRACT

Peer-to-peer (P2P) architectures have recently become a popular design choice for building scalable Networked Virtual Environments (NVEs). In P2P-based NVEs, system and data management is distributed among all participating users. Towards this end, a Delaunay triangulation can be used to provide connectivity between the different NVE users depending on their positions in the virtual world. However, a Delaunay triangulation clearly suffers from a high maintenance cost as it is subject to high connection change rate due to continuous users' movement. In this paper, we propose a new variant to the Delaunay triangulation algorithm that provides network connectivity to support P2P NVEs while dramatically decreasing maintenance overhead by reducing the number of connection changes due to users' insertion and movement. Performance evaluations show that our solution drastically reduces overlay maintenance cost in highly dynamic NVEs.

Keywords

Networked Virtual Environments, Peer-to-Peer Systems, Delaunay Triangulation.

1. INTRODUCTION

Networked Virtual Environments (NVEs) are 3-D virtual worlds in which a huge number of participants play roles, and interact with their surroundings and each other through virtual representations called *avatars*. Several application areas for NVEs exist, the most popular of which are Massively Multiplayer Online Games (MMOGs) where hundreds of thousands of concurrent players are currently reported. This challenges the scalability of currently employed client-server architectures where the only way to cope with an ever growing user population is to employ more dedicated servers. Clearly, this solution is extremely expensive to deploy, and its scalability is limited by server capacity which can become a bottleneck during peak loads.

This has recently led to focusing on peer-to-peer (P2P) architectures as an alternative design choice for building scalable NVEs. By aggregating and sharing users' resources, P2P architectures achieve high scalability in a cost-effective manner. In P2P systems, the overall system load is distributed among all participating users/nodes, which organize themselves into an overlay network in which they all have an equal role and act as both clients and servers.

A key aspect of P2P networks is the overlay topology structure that defines the way peers connect to other "neighboring" peers with whom they can interact and exchange messages. The overlay topology can be arbitrary or, alternatively, can be inspired by some application-specific semantics. In a typical NVE, a user sees only a portion of the virtual world where she/he can perform actions (i.e., moving around, manipulating objects, communicating with other users, etc.). This portion of the virtual world, commonly known as Area of Interest (AOI), is often based on geographic proximity. A user is thus only interested in the activities happening within its AOI. In this context, it is essential to dynamically organize the overlay network with respect to users' positions in the virtual world by having each user only connect to the set of neighboring users lying in her/his vicinity, and ensure that a user only receives state update messages of events happening within its AOI.

Towards this end, the well-known Delaunay triangulation [15] in computational geometry can be used to provide connectivity between the different NVE users based on geographic proximity. The Delaunay Triangulation provides several nice features that make it particularly attractive for P2P-based NVEs. First, it provides locality by connecting every node only to its geographically closest neighbors in the plane, thus enabling direct message exchange between nodes that are susceptible to interact in the virtual world. This locality feature also achieves high scalability by limiting the number of neighbors that a node is connected to, independently from the size of the network. A global behaviour is than achieved through cooperative local interactions. Finally, using a Delaunay triangulation allows distributed state management through the use of its corresponding dual structure, i.e., the Voronoi diagram [11], to partition the virtual world into regions equal to the number of nodes such that each node lies in the region that contains all the points that are closer to it than to any other node. Simply stated, a Voronoi diagram defines the region that each node has to manage, while the Delaunay triangulation determines the neighbors to which the node is to be directly connected. However, using a Delaunay

NetGames'08, Worcester, MA, USA.

triangulation in the context of P2P-NVEs clearly suffers from a high maintenance cost as it is subject to high connection change rate (known as *edge-flipping* operation) due to continuous users' movement [17]. Indeed, as users tend to move in the virtual world, the network topology should continuously adapt to users' movements such that the triangulation is kept valid. Maintenance cost can become particularly high when crowding occurs in parts of the virtual world, and a highly dynamic interaction with the surrounding takes place.

In this paper, we propose a new variant to the Delaunay triangulation algorithm that provides network connectivity to support P2P NVEs while dramatically decreasing maintenance overhead by reducing the number of connection changes due to users' insertion and movement. We achieve our goal by maximizing the region where a user can freely move without generating a flip operation, therefore reducing the message cost for maintaining a valid overlay. Our idea is simple, yet it proves very effective in reducing the overlay cost maintenance especially in highly dynamic environments as shown by our performance evaluations.

2. RELATED WORKS

A number of P2P designs have been recently proposed for scalable NVE support. A key issue common to all P2P-NVEs is dynamic topology maintenance. This implies that the network should cope with user movement through neighbor discovery and self-reorganization such that the overlay topology is kept consistent.

In [1], the authors propose a distributed NVE architecture based on the *Pastry* distributed hash table [3] and its corresponding application-layer multicast, *Scribe* [4]. The game world is partitioned into fixed-size regions, each managed by a unique *coordinator* node. The coordinator node of a region also serves as the root of a multicast tree to which all the peers lying in the region subscribe. State changes occurring in a region are subsequently multicast by the root node to all the region members. Coordinator nodes are randomly chosen and connected to each other using Pastry. [2] follows the same approach except that nodes of a region directly connect to the coordinator node of their region, thus eliminating message relay through direct connections. However, both works suffer from performance, scalability, and robustness issues inherent to server-based architectures as the coordinator can quickly become a bottleneck when crowding occurs.

[7] describes fully-distributed P2P architecture for NVEs. Each node maintains a fixed number of direct connections to its closest neighboring nodes. Neighbor discovery is however achieved through regular neighbor-list exchange, which clearly introduces a high message overhead. [8] is a combination of DHT, unstructured overlay, and neighbor-list exchange. The virtual world is partitioned into hexagonal cells, each of which is assigned a unique *master* node present within the cell. A DHT is used to maintain the hierarchical structure and assign masters to cells. A cell's master keeps track of all other *slave* nodes in its cell, and regularly exchanges this list with master nodes of the neighboring cells. Slave nodes are notified of new neighbors by their master, while other message exchange is performed directly between the slaves. This scheme might however overload master nodes, and although neighbor-list exchange is performed only between masters (compared to [7]), its frequency is still a concern. [5, 6] is another fully-distributed architecture where each node maintains direct connections to all its AOI neighbors (i.e., all the nodes lying in its AOI). In case of position change, new neighbors are discovered through mutual cooperation and notification among direct neighbors. However, the proposed solution might lead to inconsistencies where some neighboring nodes go undetected.

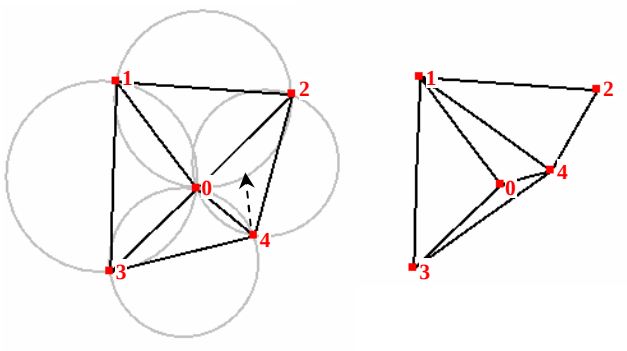
The work presented in [9, 10] discusses a Voronoi-based partitioning of space in the context of networked virtual environments. Each node maintains a Voronoi diagram of its AOI neighbors, and keeps a direct connection to all of them. Neighbor discovery is again achieved by neighbor cooperation and mutual notification. [12] and [16] are extensions of [9] that add support to data and state management in a fully distributed way. If crowding occurs, however, mechanisms that connect a node to all its AOI neighbors might lead to situations where almost all nodes become connected to each other due to overlapping AOIs (neighbor list size in $O(N^2)$, where N is the number of nodes in the system. This leads to obvious scalability problems where computing and bandwidth requirements exceed nodes' capabilities. To deal with this issue, [13] builds on the work presented in [9] to support AOI-Scalability. The work proposes to build a spanning tree across all AOI neighbors of a node, thus reducing bandwidth usage at the cost of a slightly higher latency.

However, none of the previously discussed works effectively deals with the high *maintenance* cost associated with the continuous dynamic organization of the overlay topology. This cost can become extremely high when crowding occurs, especially when it is associated with highly dynamic interactions between the nodes. As expected, these situations can be very frequent in NVEs in general and MMOGs in particular. Given this fact, [14] is, to the best of our knowledge, the first proposal to deal with the maintenance cost introduced by dynamicity and crowding. The authors describe a clustering scheme in a Delaunay-based overlay network. Every node in the network monitors its maintenance cost, and triggers a cluster creation procedure whenever its cost exceeds a given threshold. A cluster is then considered as a single node for the rest of the graph. Members of the same cluster then expand their coordinates by stretching their inter-node links. Apart from the cost associated with cluster creation and maintenance, the proposed mechanism indeed reduces the effect of density, but still performs all edge-flipping operations dictated by the Delaunay triangulation. This is exactly the issue tackled by our work. With this respect, our work can be regarded as orthogonal and both mechanisms can be combined, each bringing its own optimization.

3. DELAUNAY TRIANGULATION

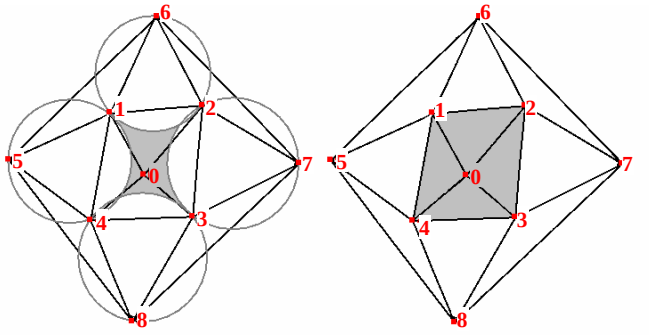
The Delaunay triangulation for a set of vertices A in a 2-D plane is a triangulation DT(A) such that no vertex of A is inside the circumcircle of any triangle in DT(A) (see Figure 1a) [15].

Note that a position change of a vertex in A can violate the basic property of DT. Let *0*, *2*, *1*, *4* be vertices in A and *021* and *024* be triangles in DT(A). If *4* changes position and enters inside the circumcircle of *021*, triangles *021* and *024* violate the Delaunay property. To meet the Delaunay condition, an *edge*



(a) Before **(b) After**

Figure 1. (a) Delaunay triangulation
(b) Flip operation resulting from node *4*'s movement



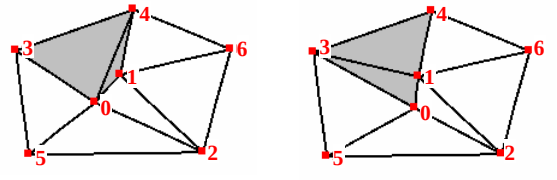
(a) **(b)**

Figure 2. Node *0*'s flip-free area in (a) Delaunay
(b) Flexible triangulation algorithm

flipping operation is performed by switching the common edge *02* for the common edge *14* resulting in two valid triangles *041* and *241* (see Figure 1). The gray area in Figure 2a shows the area where node *0* is allowed to move without triggering any flip operation.

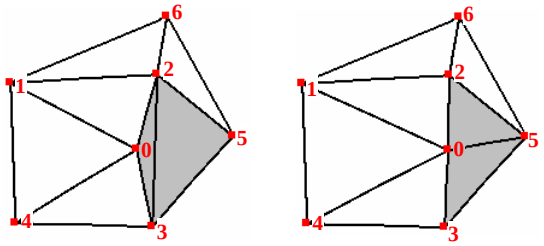
A Delaunay triangulation thus provides a connected graph of a set of vertices based on their proximity, whereby every vertex is only connected to its closest neighboring vertices in the plane. Furthermore, the average number of edges per vertex is small and independent of the size of A (generally less than six). For this reason, the Delaunay triangulation constitutes an interesting choice for the structuring of P2P-NVEs. For our purposes, every user (or *node*) in the virtual world is represented by a vertex in the Delaunay graph, with user geographical positions used as the corresponding vertex coordinates. Two nodes are direct neighbors (i.e., one-hop neighbors) in the overlay if their corresponding vertices in the Delaunay graph are connected via an edge. In the following, the terms "user", "node", and "vertex" will be used interchangeably.

To maintain a valid topology in spite of node movements, every node has to inform its direct neighbors of its position change. This allows new neighbor discovery through mutual notification where nodes collaborate to inform each others of new "approaching" nodes, and thus enables dynamic organization of the overlay through flip operations.



(a) Before **(b) After**

Figure 3. Flip operation resulting from triangle anomaly during node *0*'s movement towards node *2*



(a) Before **(b) After**

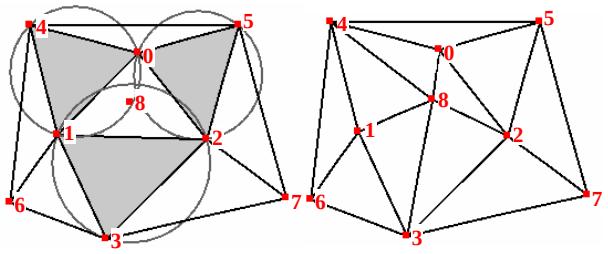
Figure 4. Flip operation resulting from crossing the base of the triangle

4. DELAUNAY TRIANGULATION REVISITED

As stated earlier, a position change in Delaunay triangulation might initiate a flip operation so that validity is maintained. NVEs are extremely dynamic environments due to continuous user movement. In order to minimize the maintenance cost resulting from high connection change rate due to user movement, we want to maximize the area where a node is allowed to move without triggering any flip operation (hereafter, we call this area a *flip-free* area). To this end, we modify the flip-free area of a node *N* to become the total of the triangular regions around *N*, i.e., the region composed of all the triangles for which *N* is a vertex. If *N*'s flip-free area is convex, then no flip operation occurs until *N* passes across the base of one of its triangles. If this area is not convex, then no flip operation occurs until either *N* passes across the base of one of its triangles, or triangle anomalies occur (see Figure 2b).

4.1 Node movement

In our approach, only two types of flip operations can occur during the movement of a node *N* inside the area containing all its triangular regions. The first type of flip operations occurs if there is an anomaly in one of *N*'s triangles. As known, no edge of a triangle can be as long as the sum of the other two edges lengths. However, *N*'s movement can violate this basic triangle rule for any of *N*'s triangles. As shown in Figure 3, the movement of node *0* towards node *2* results in the violation of this rule for triangle *014*. As a consequence, the movement of *0* causes the flip operation between triangles *041* and *043* and creates triangles *013* and *413*. This kind of flip operations occurs when the total triangular region around a node *N* is not convex. As a result of



(a) Before **(b) After**

Figure 5. Insertion of node *8* resulting in two flip operations with neighbor triangles

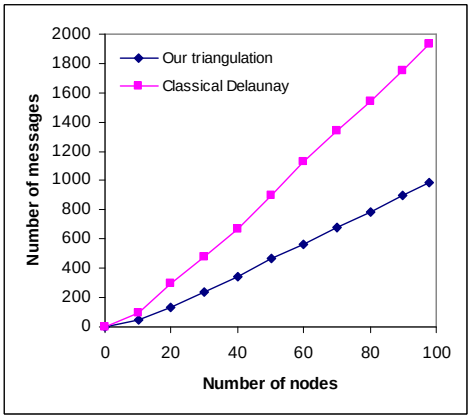


Figure 6. Total number of messages for entrance procedure

N's movement, a base of any triangle violating the convexity property of the total triangular region finishes by a flip operation switching this edge (base of triangle) with an edge that meets the convexity property.

The second type of flip operation occurs when a node passes across the base of any triangle. In Figure 4, node *0* passes across the base of triangle *023* and results in switching the edge *23* for the edge *05*. This type of flip operation adds a new triangle neighbor and so increases the total number of triangle neighbors of a node by one. In order to keep a small number of neighbors, node *N* performs a second flip operation between *N*'s two triangles which have the longest common edge, meaning that *N* does not keep anymore the connection with its farthest neighbor. In other words, a node *N* passing across the base of any triangle performs two successive flip operations. First, *N* adds the new neighbor that it gets informed about by its existing neighbors. *N* then leaves its connection with the neighbor the furthest from itself. Therefore, while adding a new nearby neighbor, *N* disconnects from its farthest neighbor. Apart from these two operation types, all other flip operations that might be triggered during a node movement in classical Delaunay are eliminated.

4.2 Node insertion

The node that wants to enter to system first contacts a gate node[1]. The gate node checks if the new node is inside one of its triangles. If so, the gate node allows the new node to enter the system according to the insertion algorithm. If not, it transmits the new node's entry demand to its closest neighbor to the new node in the virtual world, which in turn performs the same operation. If the new node is inside one of its triangles, the neighbor allows the new node to enter the system. If not, it transmits the entry message to its closest neighbor to the new node. This operation continues until the entry message reaches the node allowing the new node to enter the system.

Let *T* be the triangle inside which the new node *N* enters. *T* has three neighbor triangles with which *T* has a common edge. When *N* enters inside *T*, we also check whether *N* enters inside the circumcircle of *T*'s neighbor triangles. If so, a flip operation is done with the neighbor triangle inside the circumcircle of which *N* enters. Figure 5 shows the insertion of node *8*. Let node *0* be the node allowing node *8* to enter the system. Node *0* finds that node *8* enters inside triangle *012*. Node *0* also controls if node *8* enters inside the circumcirle of neighbor triangles *014* and *025* and so a flip operation is performed between triangles *012* and *014*. The control with the third neighbor triangle *123* is done by one of *0*'s neighbors (either node *1* or node *2*) of triangle *012*. Then the second flip operation is performed between triangles *012* and *123*.

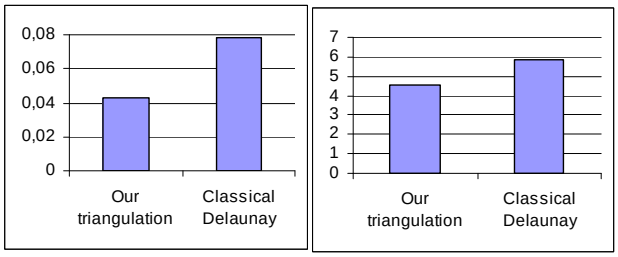
The reason behind limiting the control operation to the circumcirles of the three neighbor triangles is to avoid successive flip operations in the virtual world (as would be generated by standard Delaunay). As an example, in Figure 5, node *8* can also be inside the circumcircle of triangles *136*, *146* and *237*, meaning that several flip operations can be done causing several message transmissions between the nodes.

5. EXPERIMENTAL EVALUATION

In this section, we present results from the simulation of our triangulation algorithm. We consider a virtual world of 500x500 units. Nodes enter the world at random positions. In the first set of experiments, we evaluate the cost of node entrance according to two algorithms: our triangulation algorithm and the classical Delaunay triangulation. Figure 6 shows the total number of message transmissions during the nodes' entrance to the virtual world. In our algorithm, the entrance of a new node to the system affects at most four neighboring triangles. Therefore, our triangulation algorithm is more efficient than the Delaunay triangulation in terms of the number of messages exchanged for node joins.

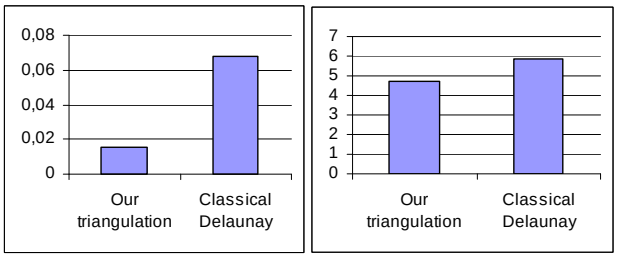
The movement of a node concerns two types of actions: informing triangle neighbors of position change, and possible flip operations. Since the communication cost of a flip operation is higher than the simple sending of a position change message to a neighbor, we evaluate separately this parameter. In the node movement simulation, a node makes a walk in the virtual world in a direction changed randomly every 30 seconds at a speed of 5 units/sec for 10 minutes during which the node is continuously moving. Figure 7a shows the number of flip operations performed per movement unit, while Figure 7b shows the number of messages sent to triangle neighbors per unit. In this simulation,

[1] We assume that there is a set of nodes, called *gate nodes*, which IP addresses are known by the system. Gate nodes allow new nodes to join the virtual world.



(a) Number of flip operations per unit **(b) Number of messages to triangle neighbors per unit**

Figure 7. Evaluation of movement



(a) Number of flip operation per unit **(b) Number of messages to triangle neighbors per unit**

Figure 8. Evaluation of movement

nodes are interested in the whole virtual world. If nodes have a stronger "locality of interest" in the nearby regions around them, and perform their movement inside this "local" environment, then our algorithm has even much better performance compared to Delaunay triangulation as shown in Figure 8.

6. CONCLUSION

Peer-to-peer (P2P) architectures have recently become a very popular design choice for building scalable Networked Virtual Environments (NVEs). While the well-known Delaunay triangulation has several nice features that make it particularly attractive for P2P-based NVEs, and thus constitutes an interesting choice to provide connectivity between NVE users based on proximity, it introduces a high topology maintenance cost as it is subject to high connection change rate due to continuous user movement in the virtual world.

In this paper, we revisited the Delaunay triangulation in the context of NVEs, and proposed an efficient variant that reduces the overlay maintenance overhead. We achieve our goal by maximizing the region where a user can freely move without generating an edge-flip operation due to users' insertion and movement, therefore reducing the message cost for maintaining a valid overlay. Our mechanism proves particularly efficient in highly dynamic environments, and achieves even better performance when crowding occurs in parts of the virtual world, and a highly dynamic interaction with the surrounding takes place. We have evaluated our mechanism through simulation, and shown that it drastically reduces overlay maintenance cost in such situations.

As a future work, we plan on evaluating our algorithm in a real setting. Another interesting research direction is to study the effect of our triangulation on the possibility of exploiting the Voronoi diagram for region decomposition and distributed data management. Indeed, in one of our previous works, we have proposed a fully-distributed object management scheme based on the Voronoi tessellation [12]. We are currently investigating how our new triangulation mechanism would affect our previously proposed study, with the objective of deriving from it a new *Voronoi-variant* that can be exploited for object management purposes.

7. REFERENCES

[1] B. Knutsson, H. Lu, W. Xu, and B. Hopkins, "Peer-to-peer support for massively multiplayer games". In *Proc. of IEEE Infocom*, pp. 96-107, March 2004.

[2] T. Iimura, H. Hazeyama, and Y. Kadobayashi, "Zoned federation of game servers: A peer-to-peer approach to scalable multi-player online games". In *Proc. of ACM SIGCOMM workshop on Network and system support for games (NetGames)*, pp. 116-120, August 2004.

[3] P. Druschel and A. Rowstron, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems". In *Proc. of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, Heidelberg, Germany, pp. 329-350, November 2001.

[4] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron, "SCRIBE: A large-scale and decentralized application-level multicast infrastructure". In *IEEE Journal on Selected Areas in Communications*, Vol. 20, No. 8, pp.1489–1499, October 2002.

[5] J. Keller and G. Simon, "Solipsis: A massively multi-participant virtual world". In *Proc. of International Conference on Parallel and Distributed Techniques and Applications (PDPTA)*, pp. 262-268, June 2003.

[6] D. Frey, J. Royan, R. Piegay, A.-M. Kermarrec, E. Anceaume, and F. Le Fessant, "Solipsis: A Decentralized Architecture for Virtual Environments". In *Proc. of International Workshop on Massively Multiuser Virtual Environments (MMVE)*, pp. 29-33, March 2008.

[7] Y. Kawahara, H. Morikawa, and T. Aoyama, "A peer-to-peer message exchange scheme for large scale networked virtual environments". In *Proc. of IEEE International Conference on Communication Systems (ICCS)*, pp. 957-961, November 2002.

[8] A. Yu and S. T. Vuong, "MOPAR: A mobile peer-to-peer overlay architecture for interest management of massively multiplayer online games". In *Proc. of International workshop on Network and operating system support for digital audio and video (NOSSDAV)*, pp. 99-104, June 2005.

[9] S. Y. Hu and G. M. Liao, "Scalable Peer-to-Peer Networked Virtual Environment". In *Proc. of ACM SIGCOMM 2004 workshops on NetGames*, pp. 129-133, August 2004.

[10] S. Y. Hu, J. F. Chen and T. H. Chen, "VON: A Scalable Peer-to-Peer Network for Virtual Environments". In *IEEE Network*, Vol. 20, No. 4, pp. 22-31, July/August 2006.

[11] F. Aurenhammer, "Voronoi diagrams-a survey of a fundamental geometric data structure". In ACM Computing Surveys, Vol. 23, No. 3, pp. 345-405, September 1991.

[12] E. Buyukkaya and M. Abdallah, "Data Management in Voronoi-based P2P Gaming". In *Proc. of IEEE International Workshop on Digital Entertainment, Networked Virtual Environments, and Creative Technology (CCNC)*, pp. 1050-1053, January 2008.

[13] J. R. Jiang, Y. L. Huang, and S. Y. Hu, "Scalable AOI-Cast for Peer-to-Peer Networked Virtual Environments". In *Proc. of International Conference on Distributed Computing Systems Workshops (ICDCSW) Cooperative Distributed Systems (CDS)*, June 2008.

[14] M. Varvello, E. Biersack, and C. Diot, "Dynamic Clustering in Delaunay-Based P2P Networked Virtual Environments". In Proc. of *ACM SIGCOMM workshop on Network and system support for games (NetGames)*, pp. 105-110, September 2007.

[15] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, "*Computational Geometry, Algorithms and Applications*", Springer-Verlag, 1997.

[16] S.Y. Hu, S.C. Chang, and J.R. Jiang, "Voronoi State Management for Peer-to-Peer Massively Multiplayer Online Games", In *Proc. of IEEE Intl. Workshop on Networking Issues in Multimedia Entertainment (NIME)*, January 2008.

[17] J. Liebeherr, M. Nahas, and W. Si, "Application-layer multicast with Delaunay triangulations", In *IEEE Journal on Selected Areas in Communications*, Vol.20, pp. 1472-1488, 2001.