Nomad: Virtual Environments on P2P Voronoi Overlays

Laura Ricci and Andrea Salvadori

Department of Computer Science
Largo Bruno Pontecorvo, Pisa, Italy
ricci@di.unipi.it,
asalvad@cli.di.unipi.it
http://www.di.unipi.it/ricci

Abstract. *Nomad* is a support for the development of *distributed virtual environments(DVE)* on P2P networks. In a DVE each avatar generally interacts with other ones in its area of interest, i.e. the DVE region surrounding its location. *Nomad* exploits a *Voronoi partition* of the DVE to define a highly dynamic P2P overlay that connects a peer with those controlling avatars in its area of interest. This paper introduces an accurate definition of neighbours and describes the basic protocol defined by *Nomad*. This protocol is then refined through a set of optimizations. Finally, preliminary experimental results are presented.

1 Introduction

In the last years, several novel applications have been added to first generation $P2P$ applications, such as file sharing and instant messaging. A new class of applications which could fully exploit the P2P model is that of *Distributed Virtual Environments (DVE)* which include both distributed simulations developed for military or civil protection purposes and massively multiplayer online games (MMOG), for instance World of Warcraft or Second Life. The definition of a scalable communication support is a basic issue for the wide diffusion of these applications in a P2P environment. Several proposals [4,5,6,7,10] exploit the concept of *Area of Interest (AOI)* to define these supports. The *Area of Interest* of an avatar in a DVE is the region of the virtual world surrounding it. Each avatar is generally aware of the avatars and passive objects in its AOI. AOI may be implemented through multicast groups or through publish subscribe systems but the scalability of these solutions is fairly low. An alternative solution exploits a fully decentralized overlay network, to support direct interactions between peers controlling avatars aware of each others. These overlays are highly dynamic because their structure may change anytime the position of an avatar changes. Direct interactions support a reduction of latency while scalability may be increased by exploiting the AOI to minimize the number of interactions. The main problem posed by this approach is to guarantee the connectivity of the P2P overlay, and to inform an avatar when another one enters its AOI from the DVE portion outside its AOI. This requires the definition of a set of 'beacon avatars' to detect new nodes entering the AOI that may be chosen among those in the AOI. However, if the AOI is empty, avatars located outside the AOI have to be chosen in order to guarantee network connectivity. A recent proposal [3,4] suggests the adoption of *Voronoi Diagrams* to partition the DVE and assign each resulting region to a distinct DVE node. In this approach, 'beacon' neighbours of a node n are chosen according to the resulting partitioning.

This paper presents *Nomad*, a P2P support for DVE based on a Voronoi approach. A node may exploit *Nomad* to join a DVE, to notify the update of its position to its neighbours, to discover new neighbours and to leave the overlay. In *Nomad* each avatar has a partial view of the DVE, hence different avatars may have inconsistent views of the DVE. Furthermore, the latency and the reliability of the transport protocol may increase these inconsistencies. As a matter of fact, a reliable connection oriented protocol is not suitable because of the high dinamicity of the overlay and of the low amount of exchanged data. *Nomad* introduces mechanisms to speed up the convergence of the partial views of the different nodes and to reduce the number of messages.

The paper is organized as follows. Sect.2 describes some recent proposals of P2P DVE. Sect.3 presents the approach based upon Voronoi Diagrams. Sect.4 describes the main *Nomad* functionalities, while Sect.5 describes the support in more details. Preliminary experimental results are shown in Sect.6 and some conclusion are discussed in Sect.7.

2 Related Work

Solipsis [5] is a P2P support for DVE that has the goal of scaling to an unbounded number of participants. Each peer implements the entities of the virtual world and perceives its surroundings. Each entity perceives only a part of the virtual world, its *Awareness Area*, inhabited by some entities and it should be aware of all updates to the virtual representations these entities. The DVE should satisfy a *Global Connectivity* property to guarantee that no entity will 'turns its back' to a portion of the world.

Mopar [6] defines an overlay network using both a Pastry DHT and an hybrid P2P architecture. It decomposes the DVE into hexagonal cells, and introduces master, slave and home nodes. Each cell has at most one master node and several slave nodes. If a new node finds that there is a cell with no master node, it registers itself as the master node. Otherwise, it becomes a slave node. Master nodes build direct connections with the neighboring master nodes through home nodes of neighboring cells. Slave nodes query the master node to build direct connections with their neighboring slave nodes. In this way, master nodes have not to be involved in the protocol to notify the accurate positions to the neighboring participants. SimMud [7] is a support for DVE built on top of Pastry, a widely used DHT, and it uses Scribe, a multicast infrastructure built on top of Pastry, to disseminate game state. The approach exploits locality of interest typical of DVE. Von [4] exploits Voronoi Diagrams [2] to assign to each node a distinct DVE region resulting from a proper partition of the virtual world. Von preserves high overlay topology consistency in a bandwidth-efficient manner. [8,9,10] describe a publish subscribe approach to the definition of DVE and introduce a set of mechanisms to guarantee the consistency of the virtual environment. An exhaustive comparison of these approaches is presented in [3].

3 Voronoi Dynamic Overlays

In the following we consider a Voronoi tessellation of the plane because 2D DVE will be taken into account. We suppose that each node of the DVE control a single avatar. As a consequence in the following the term node and avatar will be used in an interchangeable way, according to the context.

Given a set of points S in the plane, a Voronoi tessellation is a partition of the plane which associates a region $V(p)$ with each point p in S so that all points in $V(p)$ are closer to p than to any other point in S. The goal of a Voronoi based approach for DVE is to define a highly dynamic overlay network, where each node n interacts mainly with the neighbours in its *AOI*. The Voronoi approach supports a straightforward definition of the set B of 'beacon nodes' of a node n which inform n when a new node enters its *AOI*. B should be chosen in a way that pairs each region surrounding $AOI(n)$ with one node of B. In this way, each node approaching $AOI(n)$ is detected by one beacon node which then informs n. Furthermore each beacon may inform n of new neighbours discovered as a result of its movement. We can notice that the beacon nodes are not necessarily located in the *AOI* of a node because, if the *AOI* is empty, the node cannot discover new neighbours as a result of a position update. So the node looses the connectivity with the rest of the network. To solve this problem we must define a new type of neighbours, called enclosing neighbours in order to guarantee the connectivity of the overlay. An accurate definition of *AOI* and 'beacon' neighbours is required to define the *Nomad* protocol. At first, we define them by considering $Voro_{Glob}$, the Voronoi diagram including all the nodes of the DVE. Then, we update the definitions by considering $Voro_n$, the Voronoi diagram built by considering the partial view of the DVE at node n and prove that the two definitions are consistent. Let n,m be a pair of nodes in $Voro_{Glob}$.

Definition 1. *m is an* enclosing neighbour *of n iff exists a border of V(m) that overlaps a border of V(n).*

Definition 2. *m is an* AOI neighbour *of n iff* $m \in AOI(n)$.

Definition 3. *m is a* boundary neighbour *of n iff at least one of the following conditions holds.*

- $m \in AOI(n)$ *or it is an enclosing neighbour of n and* $V(m) \cap AOI(n) \neq V(m)$
- $m \in AOI(n)$, $V(m) \cap AOI(n) = V(m)$, $\exists\ k$, $k \notin AOI(n)$, *k enclosing neighbour of m, such that* $V(k) \cap AOI(n) \neq 0$.

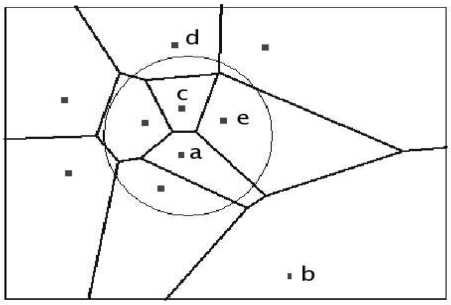

Fig. 1. Neighbour Classifications

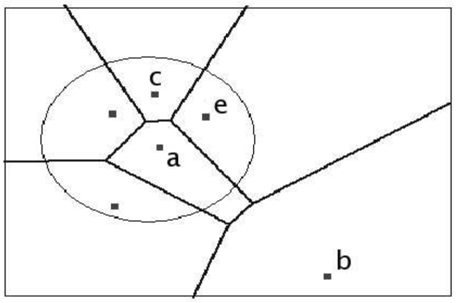

Fig. 2. Voronoi Diagram at node a

The boundary neighbours correspond to the 'beacon nodes'. According to Def.3 boundary neighbours of n are the nodes whose Voronoi region intersects the borders of the $AOI(n)$ or whose enclosing neighbours are located outside the AOI, but control a region intersecting $AOI(n)$. Enclosing neighbours of n should be considered as well, anytime the Voronoi partition does not assign some regions of the DVE surrounding n to one of the previous neighbours.

Fig. 1 shows distinct neighbours of node a. e is an AOI neighbour, because it belongs to $AOI(a)$. It is also a boundary neighbour because $V(e)$ intersects the border of the AOI. Furthermore e is also an enclosing neighbour. Even if $V(c)$ is a subset of $AOI(a)$, c is a boundary neighbour because its enclosing neighbour d is located outside the AOI, and $V(d)$ intersects the AOI. Finally, even if $V(b)$ does not intersect $AOI(a)$, b is a boundary neighbour because it is an enclosing neighbour that covers a region not assigned to other boundaries.

The final goal of Voronoi based approach is to minimize the number of nodes each node n of the DVE interacts with. Since n has a partial view of the DVE, its Voronoi diagram $Voro_n$ differs from $Voro_{glob}$, because it includes only a node subset. As an example, a partial view of n may include its AOI and enclosing neighbours. In Fig.2, $Voro_a$ differs from $Voro_{glob}$ because it either associates with AOI or with enclosing neighbours of a, some regions that in $Voro_{glob}$ belongs to nodes outside $AOI(a)$.

A simpler condition can be defined to find out boundary neighbours in $Voro_n$ in such a way that each node finds out its boundary nodes by looking at its AOI and enclosing neighbours. This condition requires to check if the Voronoi region of any AOI or enclosing neighbour of n includes at least a point outside $AOI(n)$. The following theorem shows that this condition is equivalent to the one of Def. 3.

Theorem 1. *Let m and n be two nodes, where m is an AOI or an enclosing neighbour of n. If $V(m) \cap AOI(n) \neq V(m)$ holds in $Voro_n$, then m is a boundary neighbour of n in $Voro_{glob}$.*

Proof. Let us consider n and m in $Voro_n$ such that $V(m) \cap AOI(n) \neq V(m)$ holds. Hence, $V(m)$ includes at least a point outside $AOI(n)$. If this condition holds in $Voro_{glob}$ as well, m is trivially a boundary neighbour of n because of the first condition of Def. 3. Let us now suppose that the condition is violated in $Voro_{glob}$, so that a subset of $V(m)$ is assigned to some of its enclosing neighbours which are not visible by n, i.e. are located outside $AOI(n)$. Here, m is a boundary neighbour of n because the second condition of Def.3 is satisfied.

Hence, each node n can detect its boundary neighbour by detecting in its Voronoi diagram any AOI and the enclosing neighbours whose region contains at least a point outside its area of interest. It is worth noticing that our definition of boundary neighbours updates those in [3,4] to guarantee the connectivity of the overlay.

4 The Nomad Protocol

Nomad defines a protocol to manage a Voronoi overlay. The current version of *Nomad* only supports the notification of position updates among avatars, that act as *heartbeats* notifications. We are currently extending the protocol to manage passive DVE objects. Furthermore, in Nomad, the AOI of avatars are circular region centered on the avatar, and any AOI has the same (static) radius.

While [3,4] defines a P2P Voronoi overlay, it does not investigate the definition of a support for the application level protocol on a real transport protocol. Instead, to fully exploit the overlay characteristics, *Nomad* Voronoi overlay is based upon the *UDP* transport protocol. As a matter of fact, the structure of the overlay is highly dynamic, i.e. it may change at each movement of an avatar, and a low amount of information, i.e. the avatar position and a payload with status information, is exchanged at each interaction. Hence, the choice of TCP is not appropriate, because of the cost of opening and closing dynamically a large amount of new connections. On the other hand, *Nomad* requires a set of mechanism to recover errors due to the low reliability of the underlying protocol as well as mechanisms to increase the consistency of the local views of the DVE which may differ at distinct nodes.

This section introduces the basic *Nomad* mechanisms, while the following one describes those that increase the reliability of the protocol and the consistency of the distributed views. The basic functionalities supported by *Nomad* are:

– **Join**. A new node J joins a *Nomad* overlay by notifying its initial position in the DVE and the IP address of a *bootstrap node*, i.e. any active node on the overlay, to the *Nomad* support which notifies the proper enclosing and AOI neighbours to J.
– **Position Update**. An application exploits this function when the avatar moves or to periodically send a keep alive and application payload messages. If an avatar A changes its position, the support updates the Voronoi diagrams of the node associated with A and of its neighbours.
– **Neighbour Discovery**. *Nomad* defines mechanisms to discover new AOI neighbours or enclosing neighbours of a node in the DVE.

– **Leave**. *Nomad* properly updates the Voronoi diagram of all the neighbours of a node that departs from the Voronoi overlay.

The join request is forwarded from the bootstrap node to the node A such that $V(A)$ includes the initial position of the joining avatar J, by a greedy routing algorithm [4]. That algorithm forwards, at each routing step, the join request to the neighbour closest to the final destination. A inserts J in its Voronoi diagram and exploits this diagram to discover the neighbours of J. These are notified to J, but no notification is sent to J's neighbours that can discover J through the discovery mechanisms associated to position updates messages.

Nomad periodically sends an heartbeat, including the current position of J and an optional application payload, to both the AOI neighbours and the enclosing neighbours of J. The maximal speed of J is bounded in order to guarantee that J can bypass a boundary neighbour B only after B has notified J of unknown neighbours. A heartbeat is sent also to the nodes that are no longer neighbours of J after its movement. In this way, they can discover that J is no longer their neighbour.

On the other way, no explicit request should be done at application level to discover new neighbours because the support detects them by pairing a discovery request with each Nomad heartbeat to the boundary neighbours. This is implemented by tagging the heartbeat message to these neighbours to inform them of a discovery request. When a node receives a discovery request from a node N, it looks for new neighbours of N by exploiting the received position of N and the radius of $AOI(N)$. N may receive duplicate notifications for the same neighbour, because the same neighbour notifies the same node more than one time or the same notification is received by distinct neighbours. The next section describes some strategies to reduce the number of redundant notifications. N discards duplicate notifications by checking if the notified node is already present in its Voronoi diagram and notifies the new nodes to the application. Sect. 5.2 introduce a further mechanism to discover neighbours.

Nomad can manage both voluntary peer disconnections from the overlay and unexpected peer departures. In the former case, a leave message is explicitly sent by the leaving node. Unexpected departures are managed by assigning a TTL to each neighbour, which is periodically decremented by an *aging procedure*. Furthermore, the TTL of n is set to its starting value when receiving a message from n. When the value of the TTL becomes 0, the corresponding node is eliminated from the local Voronoi diagram and it is inserted into a *Blacklist* that also includes nodes that have sent a *Leave Message*. The loss of a leave message is managed as an unexpected departure.

The *Blacklist* is exploited to avoid fuzzy situations, where a node n eliminated from the local Voronoi Diagram is immediately reinserted because some boundary neighbour still notifies the presence of n. This may happen in a fully P2P distributed environment where all the neighbours of a node cannot detect the node disconnection simultaneously. When a neighbour n is notified, it is inserted in the Voronoi Diagram if and only if it is does not belong to the blacklist. A node n can be removed from the blacklist if a notification is received from it.

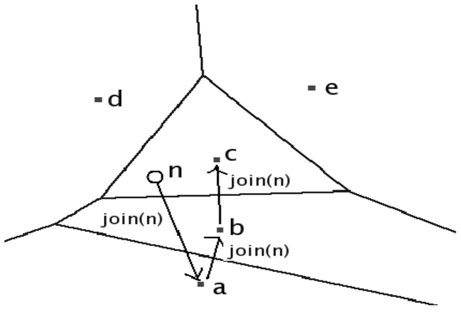

Fig. 3. Joining node n

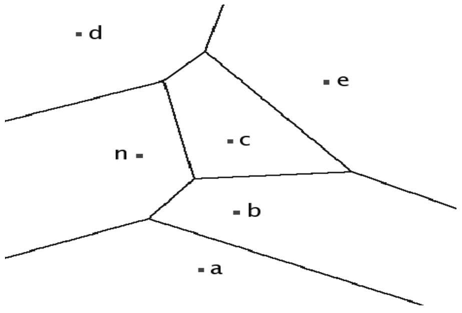

Fig. 4. Voronoi Diagram of node c

Finally, an *aging mechanism* the blacklist nodes is defied as well to limit its size and to recover erroneously detected disconnections.

5 Improving the Protocol

The protocol defined in the previous section may be improved to speed up the convergence of the local views of the DVE at distinct nodes and minimize the number of messages. *Nomad* cannot avoid any inconsistency because of both the latency of the interconnection network and the reliability of the underlying protocol. For instance, latency jitter prevents a simultaneous notification of a position update to all the neighbours of a moving avatar and notifications may be lost or delivered out of order. The number of messages to implement the protocol greatly affects the system scalability. Therefore, a set of optimizations to minimize these messages is a key point of *Nomad*.

5.1 Joining *Nomad*: Improving Initial Neighbours Discovery

In the protocol previously described, the neighbours of a node J joining the overlay are notified by the node A holding the region including the starting position of the avatar controlled by the node. Because of its limited view of the DVE, A may not know some neighbours of J. To speed up both the discovery of the neighbours of the joining node and of the new node by the other ones, *Nomad* involves in the discovery process each node that routes the join message toward its destination. When routing a join message, a node checks if the joining node J is an AOI or an enclosing neighbour. In this case, it adds J to its neighbour list, it discovers J neighbours in the updated diagram and notifies them to J. This increases the number of neighbours discovered during the joining phase and it make the protocol more robust by increasing the probability that a node joins the network even if some messages are lost.

Figure 3 shows the routing path of the join message of node n in $Voro_{Glob}$ if a is the bootstrap node and the initial position of n belongs to $V(c)$. The path includes node a, b and c. Figure 4 shows $Voro_{Glob}$ after the insertion of

n. Now the neighbours of n are a, b, c, d. Figure 3 shows that if a does not belong to $AOI(c)$, c is not aware of a and cannot inform n about its presence. n will discover a through the *Nomad* discovery mechanism associated to heartbeat notifications. In the improved protocol, when routing the message, a realizes that n is its enclosing neighbour and inform n about itself.

5.2 Implicit Detection of Neighbours

While the basic mechanism for the detection of new neighbours is based upon an explicit request to the boundary nodes, *Nomad* defines also an implicit mechanism to detect new neighbours because when receiving an heartbeat from n, a node checks if n belongs to its area of interest and, in this case, it adds n to its neighbour list. This mechanism is sound because all the area of interest have the same radius so that the relation 'belongs to an area of interest' is symmetric, i.e. if A belongs to the $AOI(B)$, B belongs to that of $AOI(A)$.

5.3 Managing Out of Order Notifications

Notifications may be received out of order because of the underlying protocols. Each instance of *Nomad* associates a *logical timestamp* with each message sent on the overlay. The timestamp is a counter incremented when sending a heartbeat. An heartbeat received by node m from n, is out of order if the value of its timestamp is less than the more recent timestamp from n. Such an heartbeat is discarded because it is obsolete. The timestamp is exploited to detect aged replies to discovery request as well. Each node receiving a discovery request with timestamp T, pairs T with the reply message, i.e. the discovery response. The sender of the discovery request can discard replies that are obsolete.

5.4 Reducing Discovery Traffic

The explicit *Nomad* discovery mechanisms sends a discovery request to the boundary neighbours which send a reply message with the new discovered neighbours. Each boundary node B may simply insert in the reply message all the nodes it has detected and that belong to the $AOI(N)$ or are enclosing neighbours of N, where N is the requesting node. This set may include nodes already known by N, because they have been previously notified by B itself or by other neighbours of N. To minimize the number of messages, *Nomad* should include in the discovery reply the nodes not known by N only. Since in a fully distributed environment the corresponding protocol is extremely complex, because of the limited knowledge of each node, *Nomad* implements a simpler strategy. Each boundary node insert in a discovery reply a node only if it has not been previously notified to the same neighbour. For this reason, a boundary node maintains, for each neighbour n, *KnownList(n)*, the list of nodes notified to n. A node may be removed from the knownlist when the neighborhood relations are updated and be re-inserted in that list later.

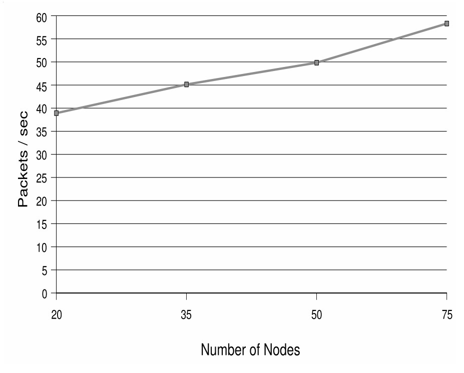

Fig. 5. Network Traffic: Packets

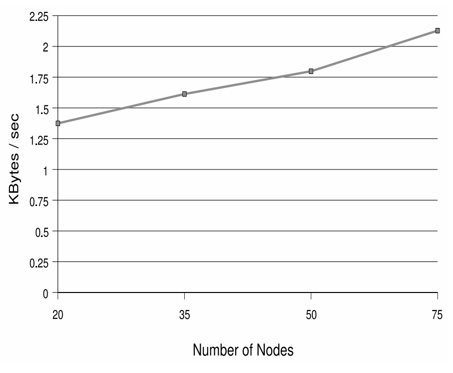

Fig. 6. Network Traffic: KBytes

6 Experimental Results

Nomad has been implemented in JAVA [1] and tested on a *LAN*. We have experimented the inefficiency of the standard JAVA serialization mechanism and we have defined an ad hoc serialization strategy reducing the number of bytes needed to code *Nomad* messages. The first set of experiments test the scalability of the support by measuring the *network traffic*, i.e. the number of packets/kbytes exchanged as a function of the number of nodes. In each experiment the network traffic has been measured for 5 minutes, starting from the instant when all the nodes have joined *Nomad*. The application is a simple game where avatars moves randomly. Each avatar chooses a random direction, then moves in that direction and changes its direction when it finds some obstacles or when a fixed period of time has elapsed. Fig.5 and Fig.6 show, respectively, the average number of packets/KBytes transmitted by a node in a second. The experiments show that the increase of sent packets/kBytes is small when the number of nodes increases. In the second set of experiments a single avatar of the *DVE* moves, while the

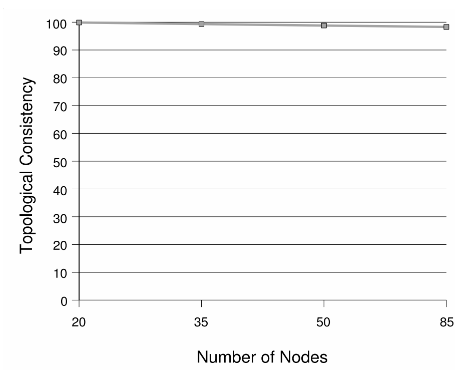

Fig. 7. Topological Consistency

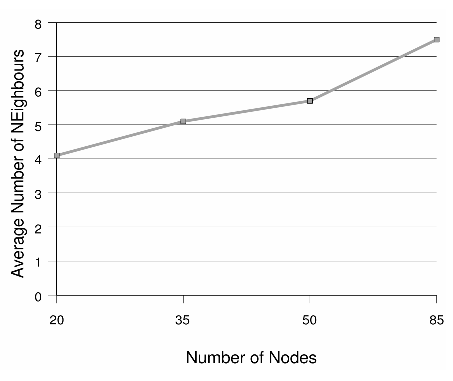

Fig. 8. Average Neigh. Number

others do not change their initial position in the DVE. This experiment measures the *topological consistency*, i.e. the percentage of *AOI neighbours* discovered by *Nomad* with respect to those really located in its area of interest. In Fig. 7 we can notice that, even if topological consistency decreases when the number of nodes increases, its value remains high (about 98 %). Finally Fig. 8 shows the average number of neighbours as a function of the number of nodes.

7 Conclusions

This paper has presented *Nomad*, a P2P support for DVE based on *Voronoi* tessellations. We have extended and refined the proposal presented in [4] in several directions. A formal definition of different kind of neighbours has been given which guarantees the connectivity of the overlay. We have developed and evaluated a prototype which speeds up the convergence of the different views of the DVE at different nodes. While the focus of [3,4] is not the choice of the most suitable transport protocol to support DVE applications, we have chosen UDP and defined a set of mechanisms to face its unreliability. We plan to extend *Nomad* to consider the management of DVE passive objects and *crowding scenarios*, i.e. situations where a high number of avatars are located in the same zone of the DVE. Finally, we are porting *Nomad* on a WAN to test the effect of latency jitter and high rates of packet loss on its behaviour.

References

1. Salvadori, A.: Nomade: Overlay Network Dinamiche basate su Diagrammi di Voronoi per Ambienti Virtuali Distribuiti, Grad. Thesis, Univ. of Pisa (April 2007)
2. Aurenhammer, F.: Voronoi Diagrams-A Survey of a Fundamental Geometric Data Structure. ACM Computing Surveys 23(3), 345–405 (1991)
3. Hu, S.: Scalable Peer-to-Peer Networked Virtual Environment, Master's thesis, Tamkang University, Taiwan (January 2005)
4. Hu, S., Chen, J., Chen, T.: VON: A scalable peer-to-peer network for virtual environments. IEEE Network 20(4), 22–31 (2006)
5. Keller, J., Simon, G.: Toward a Peer-to-Peer Shared Virtual Reality. In: Proceedings of the 22nd International Conference on Distributed Computing (July 2002)
6. Yu, A., Vuong, S.T.: MOPAR: a mobile peer-to-peer overlay architecture for interest management of massively multiplayer online games. In: NOSSDAV 2005 Washington, pp. 99–104 (June 2005)
7. Knutsson, B., Lu, H., Xu, W., Hopkins, B.: Peer-to-Peer Support for Massively Multiplayer Games. In: IEEE INFOCOM 2004, Hong Kong (March 2004)
8. Baiardi, F., Bonotti, A., Genovali, L., Ricci, L.: A publish subscribe support for networked multiplayer games. In: IASTED Internet and Multimedia Systems and Applications (EuroIMSA 2007, Chamonix, France (March 2007)
9. Baiardi, F., Genovali, L., Ricci, L.: Improving Responsiveness by Locality in Distributed Virtual Environments. In: 21th ECMS, Prague (June 2007)
10. Bonotti, A., Genovali, L., Ricci, L.: DIVES: A Distributed Support for Networked Virtual Environments. In: 20th IEEE AINA 2006, Wien, Austria (April 2006)