Hierarchical P2P Overlays for DVE: An Additively Weighted Voronoi Based Approach

Michele Albano*†, Laura Ricci*‡ and Luca Genovali§
* Dept. of Computer Science, University of Pisa
Largo B. Pontecorvo, 3, 56127, Pisa, Italy
†Dept. of Computer Science, Stony Brook University
Stony Brook, 11790, New York, USA
‡Institute for Science and Technology of Information, National Research Council
Via G. Moruzzi, 1, 56124, Pisa, Italy
§IMT: Institutions, Markets and Technologies, Institute for Advanced Studies
Piazza S. Ponziano 6, 55100, Lucca, Italy

***Abstract*—This paper presents a support for the development of *Distributed Virtual Environments* (*DVEs*) on *P2P architectures*. A hierarchical overlay is defined by pairing each peer with a *weight* which is proportional to its networking bandwidth. Peers characterized by higher weights are assigned a greater workload, in terms of connections with other peers and of number of passive objects they manage, and can act as *superpeers* that offer a set of services to peers characterized by lower bandwidth. *Additively Weighted Voronoi (AWV) Diagrams* are exploited to define a partition of the *DVE* that assigns to each peer a region whose size is dependent on the peer's weight. *Superpeers* are modeled by sites of the tessellation that have absorbed at least the Voronoi region of another site. A set of experimental results shows that this approach can be a load balancing mechanism for peer-to-peer networks, that does not impair usual properties of Voronoi-based peer-to-peer networks.**

***Index Terms*—distributed virtual environments, load balancing**

I. Introduction

Distributed Virtual Environments (DVEs)[1] such as military or civil protection distributed simulations and massively multi-player online games (MMOGs), for instance *World of Warcraft* or *Second Life*, are currently a class of challenging distributed applications, because they integrate graphics, networking and AI programming.

Even though these applications might exploit the high scalability of *peer-to-peer (P2P) networks*, most of them still follow the *client server model* because of the complexity of a fully distributed solution. On the other hand, the definition of a P2P support for DVEs is currently an active research area [2][3][4][5][6].

A DVE includes a set of *active entities*, represented by *avatars*, interacting with each other and with a set of *passive objects* like weapons, potions, etc. Each avatar is generally aware of avatars and passive objects located in its *Area of Interest (AoI)* i.e. an area surrounding it. This *Locality of Interactions* may be exploited to reduce the amount of notifications exchanged through the P2P overlay. Even if this solution may reduce the traffic on the network, it is not effective in *crowding scenarios*. In such a scenario, a huge amount of peers are located in the AoI of a peer P which should send/receive a large set of notifications to/from its neighbors. Note that in a DVE crowding may often occur, because peers may be attracted by the same object, for instance a sword or a magic potion, or may gather to fight against each other. Peers with *low bandwidth* that are not able to support this amount of traffic may experience low responsiveness for the application.

This paper defines a *hierarchical P2P overlay* for DVEs where peers characterized by a large bandwidth act as *superpeers* and give support to peers with low bandwidth which are located in their neighborhood.

Currently, *hierarchical P2P networks*, like *Gnutella 0.6*[7] and *Kazaa*[8], mainly support *file sharing* applications. In that case, the heterogeneity of peers is exploited to define a *hierarchy* where peers with a large amount of resources, such as network bandwidth, act as *superpeers* by defining a set of services for peers with fewer resources, i.e. the *ordinary peers*. Since *file sharing* is the target application of these networks, the main task of a superpeer is to index files shared by a set of ordinary peers and to forward the queries submitted by them. Each ordinary peer is generally connected to a single superpeer, whereas the superpeers exchange the queries through a *P2P overlay*. A set of distributed mechanisms is defined to *dynamically elect* the superpeers so that their number may be dynamically tuned according to the network load.

At the best of our knowledge, only a few proposals of a *hierarchical P2P overlay* for Voronoi based DVEs have been proposed until now, and none of them takes into account load balancing between peers' workload. As a matter of fact, most of the existing overlays are characterized by a *flat topology* where each peer is connected to each peer in its AoI. Our approach exploits network heterogeneity to define a *hierarchy of peers*, where peers characterized by a high bandwidth give

9781-4244-3941-6/09 $25.00

support to peers with lower bandwidth in critical scenarios, like crowding. Each low bandwidth peer should choose a high bandwidth neighbor N as *a proxy* on the overlay, so that its notifications are sent/received through N. Furthermore, the generic peer M may play a different role according to the DVE scenario, for instance M may act as a *superpeer* if the number of neighbors is under a threshold, while requiring the support of another superpeer if this number exceeds the threshold.

We exploit *Additively Weighted Voronoi (AWV) Tessellations*[9] to model the hierarchical overlay. An AWV tessellation assigns a *weight* to each site s, so that the size of its region is related to its weight. The distance between a point P and a site s, exploited by AWV tessellations to partition the points of the $2D$ space among the sites, is computed as the difference of the euclidean distance between P and s and the weight of s. The hierarchical P2P overlay is modeled by a AWV tessellation where each site corresponds to a peer and the weight of the site is proportional to the network bandwidth of the corresponding peer. A passive object o is managed by the peer whose site owns the location of o. Even if several resources may be considered when defining the weight, in this paper we will focus our attention on the communication bandwidth. The adoption of AWV Tessellations to model hierarchical P2P overlays implements a *load balancing strategy*: all other things being equal, larger Voronoi regions are assigned to peers characterized by larger weights, hence the number of passive objects assigned to each peer will be related to its weight.

The paper is organized as follows. Section II introduces some related work. Classical and weighted Voronoi tessellations are described in Section III. Section IV describes our approach, while the algorithms exploited to manage the hierarchical overlay are described in Section V. Experimental results are shown in Section VI. Finally, Section VII reports some conclusions.

II. Related Work

Solipsis[5] and *Apolo*[10] define a proper dynamic overlay for *MMOGs*. In *Solipsis*[5] nodes are self-organized into a pure peer-to-peer network, in which relationships between nodes depend on virtual proximity. In *Apolo*[10] each node connects to the closest neighbors in each of the four quadrants. To implement the multicast of a packet to all the peers in an AoI (*AoI-cast*) each node sends a message to its closest neighbors and the message is then forwarded until it reaches the borders of the *AoI*. In this way each node defines direct links with its four neighbors only. On the down side, the number of hops required to reach a peer belonging to the *AoI* might be high. *Mopar*[11] considers an alternative approach exploiting a *Distributed Hash Table*[12] to assign passive objects to the peers.

The definition of a P2P overlay for *MMOGs* based upon *Voronoi Diagrams*[9] has been investigated in [6]. In *Voronoi based* approaches, the position of each peer in the virtual world is exploited to define a *Voronoi tessellation* of the virtual world. Given n sites corresponding to n peers, a *Voronoi tessellation* partitions the virtual world into n *regions* such that the region corresponding to a site n includes all the points which *are closer* to n with respect to any other site. Two sites are *Voronoi neighbors* **iff** the borders of their regions overlap. The connected graph defined by linking neighboring sites is the *Delaunay Triangulation* associated to the *Voronoi tessellation*, and is the *P2P* overlay that is defined.

Varvello et al [17] had proposed a clustering scheme for Delaunay-based P2P NVE, where nodes monitor the cost of maintenance of the Delaunay overlay, and trigger the creation of a cluster when it exceeds a predefined threshold. The idea of an "aggregrator" in Voronoi State Management (VSM)[18] is another example where state management tasks are delegated to "arbitrators" and may be aggregated by "aggregators" when needed. Moreover, the idea that peers may host object management responsibility was described in [18]. The notification of updates to potential users who could see objects was described as the "PVR" concept in [19] (a specialized instance of the "nimbus" concept in "aura-focus-nimbus", described in [20]), where each player is assigned a "responsibility region", along with all the objects that are located in such region.

III. Voronoi Tessellations

Given a set of sites $S = s_1...s_n$, that are points in a plane, a *Voronoi tessellation* is a partition of the plane into regions, which assigns to each site s_i a region $Voro(s_i)$, that is the set of points of the plane which are closer to s_i than to any other site $s_j \in S$, according to a given definition of distance.

A *Delaunay triangulation*[9] of a set of sites S is a collection of edges satisfying an *empty circle* property. For each edge it is possible to find a circle containing the endpoints of the edge, and no other site. The Delaunay triangulation is the dual structure of the Voronoi diagram and can be obtained by connecting two Voronoi sites s_1, s_2 **iff** $Voro(s_1)$ is adjacent to $Voro(s_2)$, i.e. they share a common edge. The *Voronoi Overlay* is a peer-to-peer network where two peers are connected **iff** their corresponding sites are connected in the Delaunay triangulation.

A. Classical Voronoi Tessellation

The classical Voronoi tessellation, and hence the classical Delaunay triangulation, is defined by considering the standard L^2 metric for distance:

$$d = ||s_i, s_j|| = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

where (x_i, y_i) are the coordinates of site s_i, and (x_j, y_j) are the coordinates of site s_j. Figure 1 shows an example of a classical *Voronoi tessellation*.

B. Additively Weighted Voronoi Graph

Distances different from the one induced by the standard L^2 metric, can be exploited to build a Voronoi Tessellation from a set of sites S. For instance, it is possible to assign a *weight* w_i to each site s_i, to tune the size of its Voronoi region $Voro(s_i)$. The resulting tessellation is referred to as *Weighted Voronoi Diagram*[9].

The most common weighted distances are the *multiplicatively weighted* one:

$$d^m = ||s_i, s_j||/w_i = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}/w_i$$

and the *additively weighted* one:

$$d^a = ||s_i, s_j|| - w_i = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} - w_i$$

that lead respectively to the *Multiplicatively Weighted Voronoi Graph* and to the *Additively Weighted Voronoi Graph*. *Additively Weighted Voronoi Graphs, (AWVs)* partition the plane into a set of connected regions, where the sides of the regions are *hyperbole arcs*. In general, sites with larger weights own larger regions.

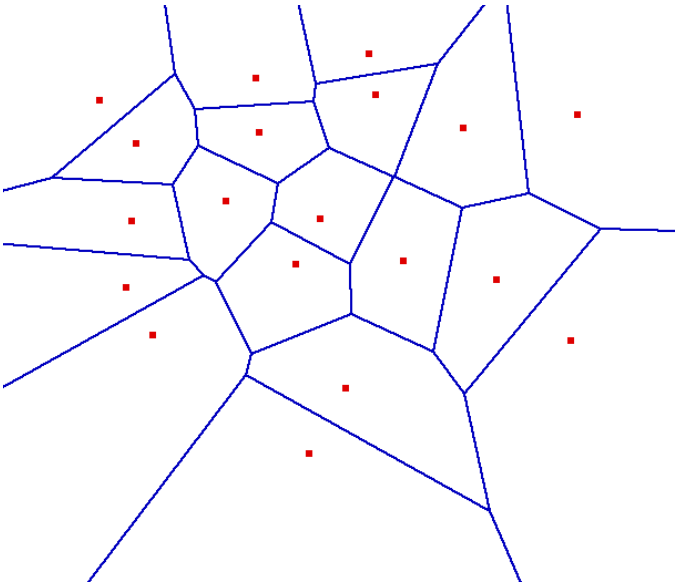

Fig. 1. Classical Voronoi tessellation.

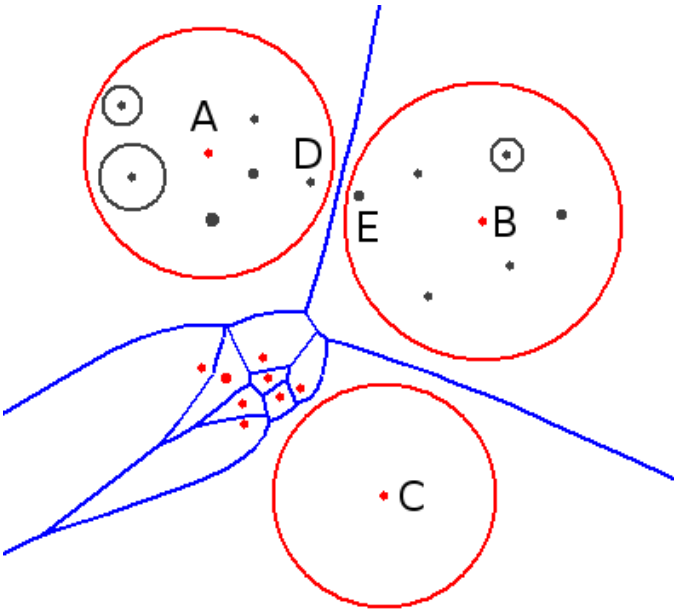

Fig. 2. Additively Weighted Voronoi tessellation.

Figure 2 shows a *AWV diagram* that includes both *visible* and *hidden sites*. The former ones own Voronoi regions which may include a set of hidden sites. The latter do not own any region and do not belong to the Delaunay Triangulation. The weights of the peers are shown by circles centered at the corresponding peer having radius that is proportional to the weight of the peer. As a matter of fact, a site s_i with an high weight may 'absorb' the region of a close site s_j with a low weight.

If all the sites are assigned the same weight the AWV graph degenerates into a standard *Voronoi graph*. Note that the extent of the Voronoi region of a peer and the number of hidden peers assigned to it, depend both on the ratio between its weight and the weight of the peers located in the neighborhood, and on the distance from neighboring peers.

IV. Modeling Hierarchical Overlays by AWV Tessellations

A *Hierarchical P2P Network* can be modeled by means of a *Weighted Voronoi Tessellation* where each site corresponds to a peer, whose *weight* is proportional to the bandwidth of the peer. Even if several resources may be considered when defining the weight, in this paper we focus our attention on the communication bandwidth only.

Visible sites model peers that do not rely on the support of a *superpeer* to forward/receive their notifications. These peers may be characterized by an high weight, or they have low weight, but are not able to find out a superpeer in their surroundings that is able to support them. Hidden sites model peers that do rely on the support of a superpeer, that is the peer that has 'absorbed' their Voronoi region. The superpeer acts as a *proxy* for the hidden peer, so that any notification is sent/received through it.

Figure 2 shows an *AWV tessellation*. In the example, both peers A and B have absorbed some neighbors, that are shown within their Voronoi regions. Figure 2 also shows a crowd of peers characterized by low weights, that have not been absorbed by any superpeer, because they are far away from high weight peers. As we will discuss, this situation may be handled by *dynamically adapting* the weights of the peers.

Let us now consider the management of the *passive objects*. Each object is dynamically associated to the peer owning the Voronoi region where the object is currently located. This peer will be called the *owner* of the object, it stores the object's state and manages concurrent updates. The ownership of an object may be delegated to another peer, when the owner moves away and the object is included in the Voronoi region of another peer. Our approach naturally supports a *load balancing* strategy, because peers characterized by larger weights are associated to larger Voronoi regions, hence they manage a larger number of passive objects.

An important issue in our approach is the definition of proper weights for the peers. A simple solution *statically assigns* weights to peers, according to their bandwidth. For

instance, it is possible to define two classes of peers, the *heavy* peers, and the *weightless* peers, and they are respectively assigned an high weight and a weight equal to zero. In this way, a *weightless* peer never hides other peers, while the *heavy* peers may act as superpeers in crowding scenarios. The weight assigned to *heavy* peers must be carefully chosen with respect to the size of the DVE. As a matter of fact, if it is too high, a few superpeers would serve all the other ones, so reverting to a classical client server architecture. If the weight is too small, the architecture would not implement any load balancing strategy in a crowding.

A more sophisticated solution is based on the *dynamic adaptation* of the weights. For instance, a superpeer should *reduce* its weight when its bandwidth is saturated by active connections, while it should *increase* its weight when its bandwidth is not fully exploited. Consider for instance the peer C in Figure 2, which is characterized by a high weight, represented by the circle centered at C. Since it has not absorbed any peer and the number of its neighbors in the Delaunay overlay is small, its bandwidth is not fully exploited. Therefore C should dynamically increase its weight in order to give support to some peers of the crowd.

V. AWV Based Overlays Management: Distributed Approaches

Basic algorithms for the distributed management of a Voronoi based overlay have been presented in [2][3][4]. This Section briefly reviews the most important concepts and describes the additions required to support the *AWV* extension.

The first step performed by a new peer N joining an existing Voronoi overlay is the discovery of its Voronoi neighbors in order to define a set of initial connections with them. N notifies its initial position I to a *bootstrap peer*, which forwards the request by *greedy routing* to the peer P owning the Voronoi region including the point I. In this way, P puts N in touch with its neighbors, and the initial connections can be defined. These connections are going to change during the permanence of N in the DVE, because N moves around the world and hence new Voronoi neighbors might be defined.

In a *DVE* each peer periodically sends a *heartbeat*, i.e. a message notifying its position, to all the peers in its *Area of Interest (AoI)*. These messages may be forwarded through the Voronoi links by proper routing strategy which exploits the properties of the Voronoi tessellation to minimize the amount of messages exchanged. [13] proposes *compass routing* to dynamically define a *spanning tree* covering the peers in the AoI. To reduce the notification delay a set of direct connections between a peer and a subset of peers belonging to its AoI may be defined.

When a peer P receives an heartbeat from Q, it checks if one of its neighbors, say N, is entering the Area of Interest of Q, and, in this case, it notifies to N the position of Q. In this way, each peer acts as a *beacon* for each neighbor by putting it in touch with new peers approaching its Area of Interest from far away locations. A similar approach may be adopted to notify the updates of passive objects. In this case, the notification is sent to each peer belonging to the *visibility area* of the object, which is generally a circular area centered in the object. These notifications may be forwarded via the same routing mechanism defined for heartbeats.

Let us now consider the *Additively Weighted Voronoi* approach. The routing algorithms proposed in [3] for the basic Voronoi approach may be exploited to forward notifications among the visible peers. A different strategy must be defined to route the notifications generated by peers which are hidden by the Voronoi regions of other peers. These notifications should be sent to the superpeer SP, i.e. the peer which has 'absorbed' the hidden peer. SP, in turn, dispatches these notifications to its hidden peers which are interested in these notifications, and to its visible neighbors. It is worth noticing that peers which are physically close in the DVE may be hidden by different superpeers, like D and E in Figure 2. For this reason each notification received by a superpeer must be forwarded to its visible neighbors which, in turn, may propagate the notification to interested peers hidden in their regions. This basic mechanism can be improved by propagating on the overlay only notifications of hidden peers whose AoI intersect the border of their superpeer's region.

It is worth noticing that each peer may turn its state from hidden to visible and the other way around, when moving within the DVE. Each peer is able to autonomously decide if it has been 'absorbed' by a superpeer: if the point the peer s_i is located does not belong to the peer's region, i.e. if there is a peer s_j such that $||s_i, s_j|| < w_j - w_i$, peer s_i is absorbed by peer s_j. When a peer is absorbed, it has no region assigned to it and it has no Voronoi neighbors, hence it resets all its connections to visible peers and establishes a single connection with its superpeer. On the other hand, a hidden peer P cannot autonomously decide if it has become visible, since it does not know any peer of the DVE, except its superpeer SP. In this case, when SP hides no more P, it notifies to P the change of its status. P receives from SP the list of its visible neighbors as well. Note that this operation is equivalent to a new join of the hidden peer to the Voronoi overlay.

VI. Experimental Analysis

The model proposed in Section IV has been analyzed by means of simulations. The simulator is composed by stable and mainstream components. A mature library that provides AWV graph's creation and management is CGAL (Computational Geometry Algorithms Library)[14], written in C++ and developed by a large consortium of European research institutions. The simulation infrastructure is Peersim[15], a very scalable event driven simulator developed in Java by researchers of the University of Bologna and used by a very large community. The softwares are linked together by means of a compatibility layer created using SWIG (Simplified Wrapper and Interface Generator)[16].

The experiments were aimed at the analysis of the topology of peer-to-peer networks based on AWV graphs, under the *statically assigned weights* scenario introduced in Section IV. The set of simulations is composed by different simulation

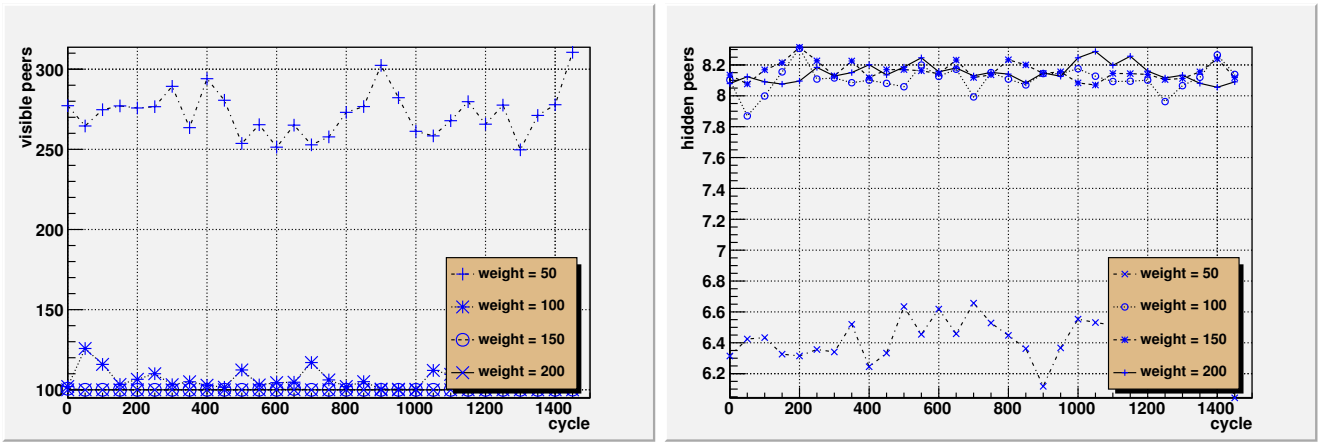

Fig. 3. Number of visible peers (left) and mean number of hidden peers per superpeer (right) for different weights.

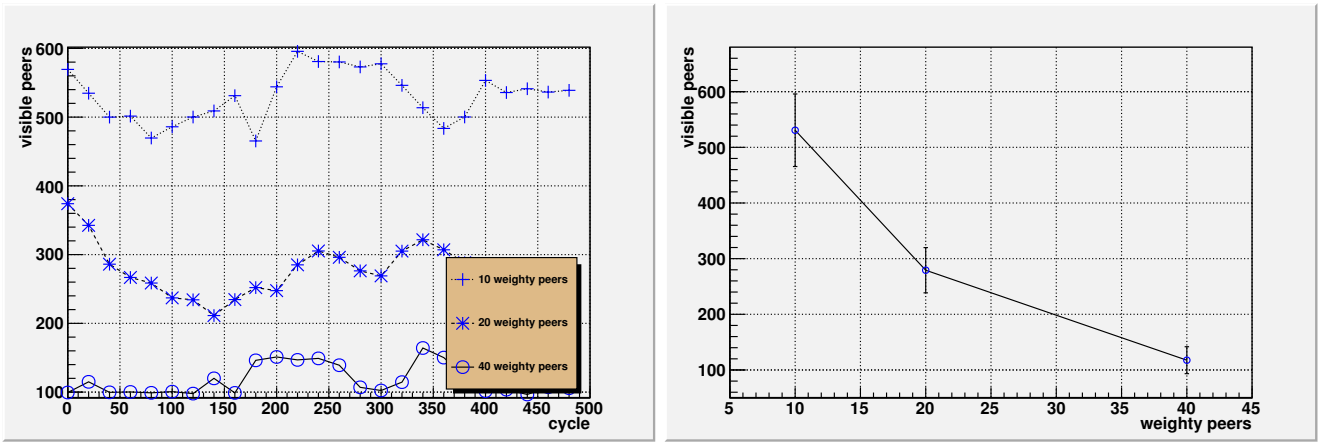

Fig. 4. Number of visible peers (per cycle and mean value) for 10, 20 and 40 *heavy* peers.

runs, and it features *weightless* and *heavy* peers, with all the *heavy* peers having the same weight p that depends on the simulation run. Some of the runs feature also passive objects. All the runs comprise a number of simulation cycles. At the beginning of each run, each peer and each passive object are assigned coordinates chosen uniformly at random in the DVE. The DVE is a rectangular area of size 600x800, and at the beginning of each cycle each visible and hidden peer 'moves itself' by a step of length 1 in a random direction.

1) Constant number of peers, different p: These simulations comprise 4 different runs, with p respectively set to 50, 100, 150 and 200. The peer-to-peer networks are composed by 800 *weightless* peers, 100 *heavy* peers, and no passive objects. Peersim simulator performed 1500 cycles for each simulation run. Left side of Figure 3 shows the number of visible peers against the cycle number. Each line represents the outcome of the experiment for a different run (different p). The figure shows that, when $p \geq 100$, only 100 peers (the *heavy* ones) are visible. This result is also confirmed by the right side of Figure 3, that shows the mean number of peers that are hidden by each superpeer, against the cycle number. The figure shows that, when $p \geq 100$, the mean gets very close to 8. Composed with the fact that all the *weightless* peers are hidden, this means that nearly each *heavy* peer is acting as a superpeer for some hidden peer.

2) Different number of heavy *peers:* These 3 runs of simulations analyze the effects of having a small number of *heavy* peers in the peer-to-peer network. The simulated networks feature 900 peer and no passive objects, the weight p is 100, there are $n \in 10, 20, 40$ *heavy* peers in the different runs, and Peersim simulator performed 500 cycles for each simulation run.

Left side of Figure 4 shows the number of visible peers for a different number of *heavy* peers, and it is summarized by the right side of Figure 4, that shows the mean number of visible peers against the number of *heavy* peers. In principle, each *heavy* peer brings along a circle where *weightless* peers can not own a region and can not be visible, hence more *heavy* peers should lead to a smaller number of visible peers, and the results of the experiment confirm that. In fact, when there are 10, 20 and 40 *heavy* peers, there are respectively $500 - 600$, $280 - 380$ and $100 - 150$ visible peers, hence the

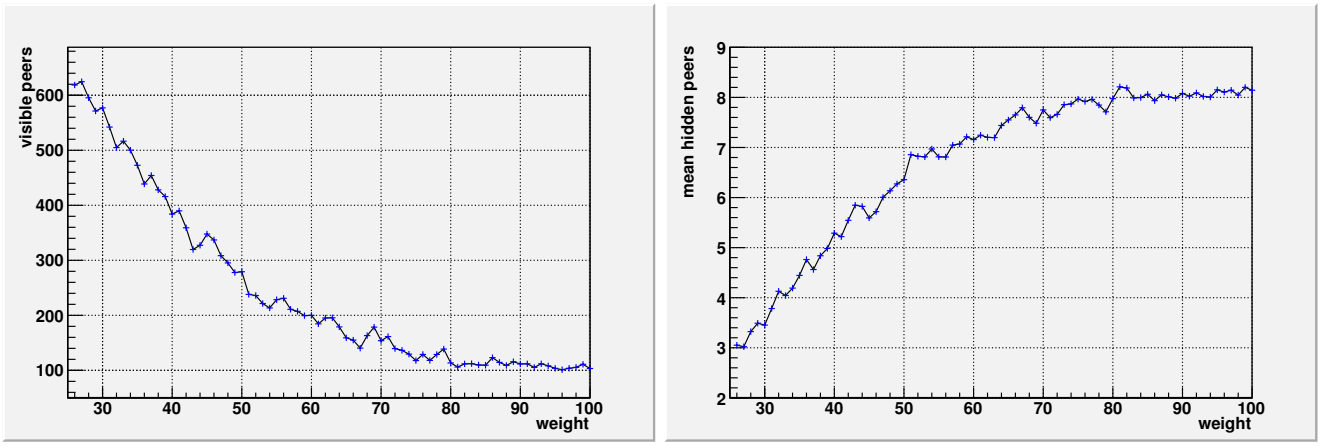

Fig. 5. Number of visible peers (left) and number of peers hidden by each superpeer (right) against the weight of *heavy* peers.

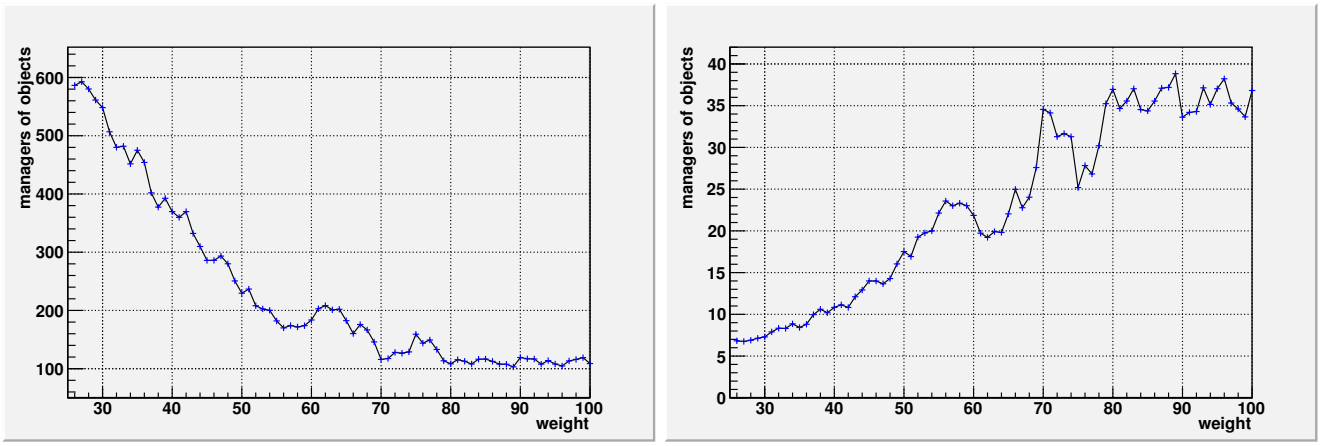

Fig. 6. Number of peers that own passive objects (left) and mean number of passive objects per owner (right).

heavy peers hide respectively $300-400$ peers, $500-600$ peers, and $750-800$ peers.

3) Active peers and passive objects: These simulations, composed by 1 simulation run, feature 800 *weightless* peers, 100 *heavy* ones characterized by the same weight p, and 4000 passive objects, to be managed by the active entities owning the object location. The simulation run involves 1500 cycles and the weight p is modified during the simulation, with $p = 100$ for the first 20 cycles, then $p = 99$ for cycles $21-40$ and so on, down to $p = 26$ for cycles $1481-1500$.

Left side of Figure 5 shows the number of visible peers against the weight assigned to the *heavy* ones. When $p \geq 80$, only the 100 *heavy* peers are still visible. This result is confirmed by the right side of Figure 5, that shows the mean number of peers that are hidden by each superpeer, against the weight assigned to the *heavy* peers. When $p \geq 80$, the 800 *weightless* peers are hidden by the 100 *heavy* peers, and hence each superpeer hides a mean of 8 hidden *weightless* peers.

Left side of Figure 6 shows the number of peers that own passive objects against the weight p of the *heavy* peers. When $p \geq 80$ the 100 *heavy* peers own all the objects, since the 800 *weightless* peers are hidden by the 100 *heavy* peers. On the other hand, when p is small, more peers (both *heavy* and *weightless* ones) own passive objects. The dual view of this result is provided in the right side of Figure 6, that shows the mean number of objects that are managed by each owner, against the weight p of the *heavy* peers. When $p \geq 80$, the 100 *heavy* peers are the only visible ones, hence they own all the 4000 passive objects, and the mean number of passive objects for each owner is $4000/100 = 40$. On the other hand, when p is small, more peers own passive objects, and hence the mean number of objects for each peer drops towards 5.

4) Number of links of visible peers: This set of simulations is composed by 1 simulation run featuring 800 *weightless* peers, 100 *heavy* ones, no passive objects, $p \in [1, 100]$. The simulation run involves 2020 cycles and the weight p is modified during the simulation, being $p = 100$ for the first 20 cycles, then $p = 99$ for cycles $21-40$ and so on, down to $p = 1$ for cycles $1981-2000$ and $p = 0$ for cycles $2001-2020$.

Left side of Figure 7 shows the mean number of links between visible peers. The number of links does not vary when the weight p is modified. More information about the number

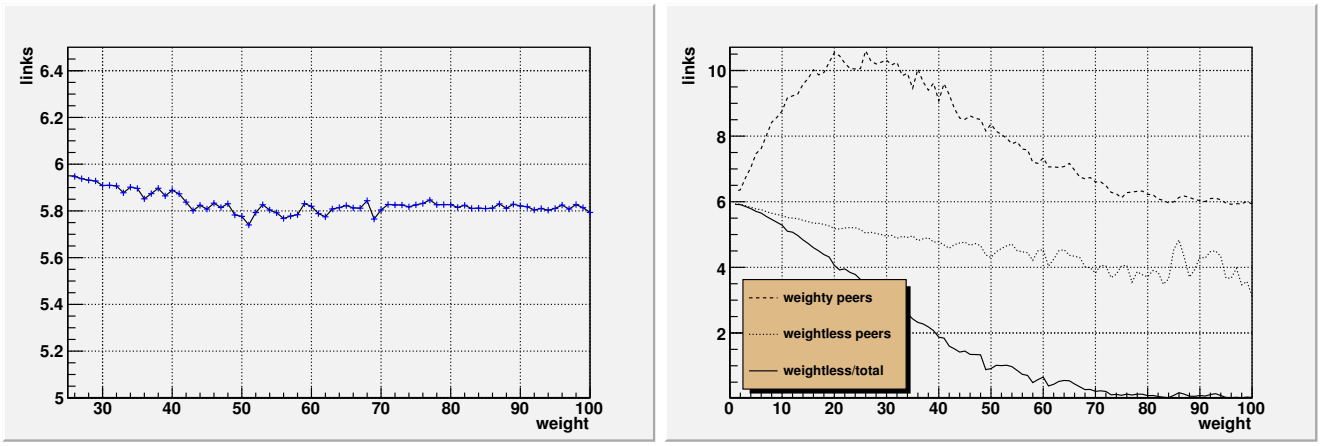

Fig. 7. Number of links between visible peers: mean value (left) and discerning *weightless* and *heavy* peers (right).

of links between visible peers is given by the right side of Figure 7, that discerns between *heavy* and *weightless* peers. The lines represent the number of links between visible peers. In particular, the higher line shows the mean number of links from *heavy* peers to visible peers. The middle and bottom line both show the number of links from *weightless* visible peers to visible peers. In particular, the middle line shows the mean number of links from *weightless* peers to visible peers, normalizing it against the number of visible *weightless* peers, completely *ignoring* hidden *weightless* peers. On the other hand, the bottom line shows the mean number of links between *weightless* peers and visible peers, normalizing it against the number of all the *weightless* peers (both visible and hidden), *counting* hidden peers as having 0 links.

This last set of simulations highlights a number of facts. First of all, the behaviors of *weightless* and *heavy* peers converge when the weight p of the *heavy* peers gets to 0, and in particular the number of links converges to 6, as it is the expected behavior for standard Voronoi graphs[9]. Moreover, the behavior of the *heavy* peers gets back to the standard Voronoi graphs' one when $p \geq 80$, meaning that the *weightless* peers are totally hidden by the *heavy* ones, and hence the remaining *heavy* peers create a standard Voronoi network between them. An unexpected yet interesting behavior is the intermediate one, when the number of links of *heavy* peers gets over 10. Some *weightless* peers, for $p \in [10, 50]$, are still visible into the network, but have small regions assigned to them, hence the *heavy* peers still have their Voronoi links between them, but they also get additional links to the *weightless* peers.

VII. Conclusions & Future Works

This paper presents a novel architecture for peer-to-peer networks, based on Additively Weighted Voronoi (AWV) graphs. This architecture is suitable for Distributed Virtual Environments (*DVEs*) and can manage an heterogeneous network that includes peers characterized by different resources, e.g.: network bandwidth. Moreover, passive objects are assigned to peers such that the management load for the peers gets balanced. The paper highlights the characteristics of this kind of peer-to-peer networks and architecture, and evaluates this solution.

We plan to develop an implementation on a real platform and to evaluate the effectiveness of our solution in real environments. An alternative to the AWV graph is represented by Multiplicatively Weighted Voronoi graphs, and we plan to evaluate this solution as well. Another open issue is the simulation of the mechanisms for the migration of the passive objects between the peers, and the issues that arise from topology consistency between the view of peers in a dynamic environment. Simulations can also profit from mobility models that are more realistic than random walks, and we are turning our attention towards group mobility models, like *Nomadic Community Mobility Model* and *Reference Point Group Mobility Model*, described for example in [21].

References

[1] S. Rueda, P. Morillo, J.M. Orduna, J. Duato *On the Characterization of Peer-To-Peer Distributed Virtual Environments*, Proceedings of the Workshop on Principles of Advanced and Distributed Simulation (PADS'05)

[2] L.Ricci, A. Salvadori *Nomad: Virtual Environments on P2P Voronoi Overlays*, First Int. Workshop on Peer to Peer Networks (PPN 2007), Vilamoura, Portugal, LNCS Vol. 4803, November 2007

[3] L.Genovali, L.Ricci *AOI-Cast Strategies for P2P Massively Multiplayer Online Games*, 5th IEEE International Workshop on Networking Issues in Multimedia Entertainment (NIME'09) Las Vegas, Nevada, USA January 2009

[4] L. Genovali, L.Ricci *JaDE: A JXTA Support for Distributed Virtual Environments*, 13th IEEE Symposium on Computers and Communications Program, Marrakesh, July 2008

[5] J. Keller, G. Simon *Toward a Peer-to-Peer Shared Virtual Reality*, 22nd International Conference on Distributed Computing, July 2002

[6] S.Hu, J.Chen, T.Chen *VON: A scalable peer-to-peer network for virtual environments*, IEEE Network, vol. 20, no. 4, Jul/Aug. 2006.

[7] *Gnutella 0.6: Protocol Specification*, rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html

[8] J. Liang, R. Kumar, K.W. Ross: *The Kazaa Overlay: A Measurement Study*, Computer Networks (Special Issue on Overlays), 2005

[9] F. Aurenhammer *Voronoi Diagrams-A Survey of a Fundamental Geometric Data Structure*, ACM Computing Surveys, Vol. 23, n. 3, September 1991, pp. 345-405

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

[11] A. Yu, S. T. Vuong *MOPAR: a mobile peer-to-peer overlay architecture for interest management of massively multiplayer online games*, NOSSDAV 05, Washington, June 2005, pp. 99-104 .

[12] I. Stoica, R. Morris, D. Karger, F. Kaashoek, H. Balakrishnan *Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications Proc*, ACM, SIGCOMM '01, San Diego, CA (2001)

[13] E. Kranakis, H. Singh, J.Urrutia *Compass Routing on Geometric Networks*, Proc. of 11th Canadian Conference on Computational Geometry, CCCG-99, pages 51-54, Vancouver Aug. 15-18 (1999)

[14] *Computational Geometry Algorithms Library*, http://www.cgal.org/

[15] The Peersim Simulator http://peersim.sourceforge.net/

[16] *Simplified Wrapper and Interface Generator*, http://www.swig.org/

[17] M. Varvello, E. Biersack, C. Diot *Dynamic Clustering in Delaunay-Based P2P Networked Virtual Environments* Proc. NetGames, Sept. 2007, pp. 105-110

[18] S.Y. Hu, S.C. Chang, J.R. Jiang *Voronoi State Management for Peer-to-Peer Massively Multiplayer Online Games* Proc. 5th Annual IEEE Consumer Communications and Networking Conference (CCNC), 4th IEEE Intl. Workshop on Networking Issues in Multimedia Entertainment (NIME), pp. 1134-1138, Jan. 2008

[19] E. Buyukkaya, M. Abdallah *Data Management in Voronoi-based P2P Gaming* Proc. IEEE International Workshop on Digital Entertainment, Networked Virtual Environments, and Creative Technology (CCNC 2008), Jan. 2008

[20] S. Benford and L. Fahln *A spatial model of interaction in large virtual environments* Proc. ECSCW, 1993, pp. 109 124

[21] T. Camp, J. Boleng, V. Davies *A Survey of Mobility Models for Ad Hoc Network Research* Wireless Communications and Mobile Computing, vol. 2, n. 5, pp. 483 - 502 (2002)