Managing Networks of Mobiles Entities using the HyVonNe P2P Architecture

Vittoria Gianuzzi, Alessio Merlo
DISI, Università di Genova
16146 Genova, Italy
Email: {gianuzzi, merlo}@disi.unige.it

Andrea Clematis, Daniele D’Agostino
IMATI-CNR, Via De Marini 6
16149 Genova, Italy
Email: {clematis, dago}@ge.imati.cnr.it

Abstract—Voronoï diagrams and Delaunay triangulations are gaining attention in several P2P applications managing a wide number of distributed and mobile entities, from resource discovery in auction-like networks to the realization of networked virtual environments. In this paper we consider a scalable partitioning technique of the entity space based on Voronoï diagrams, useful for a wide variety of distributed applications that exhibit a dynamically changing topology.

Using the HyVonNe (Hybrid Voronoï Network) architecture, the space is partitioned in Voronoï regions, each one including a limited number of entities and managed by a Region Leader, that are created and deleted depending on the spatial density of such entities, while the Delaunay triangulation connecting the Region Leaders is used to support the partitioning, routing and searching activities. The resulting two-layers structure (entity space and Voronoï regions) is scalable and extendable, allows to reduce the propagation of the entity position updates in the network and to maintain the load balancing among regions.

Simulation results related to two different application fields are presented.

I. INTRODUCTION

Peer-to-Peer (P2P) is an emerging technology for supporting a wide variety of applications. Traditionally, it is used to allow the search in large distributed archives of files, but beyond the file sharing, P2P paradigm is being used in different application fields, namely: Networked Virtual Environment (NVE) such as Massively Multi-player Online Games (MMOGs) or combat simulations, and auction-like networks.

These applications have different requirements with respect to those requested for file sharing. In NVEs the principal interest is to manage a massive number of entities located in a 2 dimensional space, each one interested to its visibility space, called Area of Interest (Aoi), represented by the space around a player and the static or dynamic entities that populate it. Spatial range queries, as well as so called window queries or nearest neighbor discovery query, are the typical requests.

In action-like networks each node publishes its offers (the entities of the network) characterized by a tuple of numerical values that locate the entity in the space. Range queries and Routing algorithms are needed to deal with the client requests.

A common feature of such highly distributed networks is that entities exhibit a dynamic or kinetic behavior: they can frequently join the network, leave or move through the space.

The most known P2P search techniques are based on fully decentralized Distributed Hash Tables (DHTs) that consider only one attribute per peer and support the exact match query needed to find well identified files in a distributed repository.

DHT based P2P networks are not able to handle the previously described applications with the low latency required, and they have been designed to handle peers almost stationary, then recently, hierarchical or graph based spatial data structures have been proposed, to support the management of such kind of networks in a P2P way, that exploit the localization of the entities. Among them, the Voronoï diagram and the related Delaunay triangulation ([1]) have been proposed, due to their properties: the Delaunay graph is always connected, greedy routing guarantees to reach the destination nearest entity, and locality aware connectivity is enabled.

However, even if only local changes are needed to maintain the Delaunay graph when a node moves, in dynamic networks the overhead of frequently updating triangulations could be very high. Clusterization techniques obtained using hybrid data structures can help to decrease the number of updates, reducing the number of movements and updates necessary to maintain the topological properties of triangulation. An important requirement in this case is the possibility of maintaining load balancing over the clusters, either when the area is crowded or it is uninhabited.

In this paper we present the design of HyVonNe, a Hybrid Voronoï Network architecture that supports efficient range and window queries even with a high churn rate of the entities of the network, as well as efficient routing algorithm. The n-dimensional space where the entities are defined is partitioned in Voronoï regions that adapt dynamically their number and boundaries so that to manage each one a limited number of entities, enabling load balancing. Each region has a virtual Region Leader (RL), that is a process running on a cooperating peer, positioned at the Voronoï site, that accepts join and leave requests of the entities, maintains the communications using a channel with each neighbour, supports routing and spatial queries, and coordinates the region partitioning together with the other leaders in a fully distributed manner.

The resulting architecture is an overlay network composed by a Delaunay triangulation over the region leaders and exhibits suitable features such as scalability, extendibility, load balancing and fault tolerance.

In the following, after a brief introduction of related works, the HyVonNe architecture will be presented and the main-
tenance algorithms will be discussed, taking in mind the synchronization features required by the concurrent execution environment. Considerations with respect to the possible routing algorithms over HyVonNe are made in Section V and, finally, two different environments are considered, and simulation results are presented, in order to emphasize the positive aspects of the HyVonNe architecture.

II. RELATED WORKS

Three approaches exist for the information management in a distributed environment: a centralized index, suitable when the number of components is small, a completely decentralized index, where each component is linked to some others in order to allow the exploration of the network, and finally the hybrid approach based on a clustering. In the last case the components are partitioned in clusters, whose heads are linked together and maintain an index of the components belonging to their cluster. It is an acceptable compromise if load balancing, and other suitable optimizations are possible, such as bandwidth reduction for communication.

Until now, in MMOGs applications, the client/server approach has been preferred, but recently P2P-based solutions have been explored, using hierarchical structure or graph to link together the peers.

Quadtrees are considered an interesting hierarchical structure when managing spatially related data. In [2], for example, the online player community is organized into a quadtree architecture, dynamically managed: the branches of quadtree are extended or retracted depending on the distribution of the online players, in order to reach the load balancing among local servers. In fact, the map can be split in smaller regions when there are too many players, while a merge is performed when they are too few. The focus of this paper is very similar to our proposal, since it combines P2P model and client-server model to obtain scalability and load balancing. HyVonNe architecture is however more flexible since it is based on a dynamic graph, non depending on a strict spatial partitioning.

A similar point of view is presented in [3] where an exagonal partitioning of the space is considered, even if the implementation is discussed considering a cluster of servers instead of a P2P network. In both cases the partitioning is performed dividing the geographical space in a hierarchical way, while HyVonNe considers the entities space.

With respect to the NVEs, a solution for the windows query is researched in [4], where a structure, the Voronoi-based Overlay Network (VON), is discussed. The graph is built directly over the participant nodes and an implementation of the system is also available. VON maintains a P2P topology exploiting the locality of the user, and achieves high scalability by bounding resource use directly at each participating node.

A distributed algorithm for the clustering of peers in an NVE that are organized using a P2P network based on the Delaunay triangulation is presented in [5]. The clustering is performed using the concept of long-short link, that is, additional edges are added to a node to link it with the closer neighbors with respect to a density function. The resulting structure is not a partitioning of the network, instead it belongs to the completely distributed index approach, since the Delaunay triangulation is performed over the node set.

In [6] the authors use Voronoi diagrams to partition the entity space into regions. Participant nodes are partitioned in simple player, and special player that act as region coordinators. Simple player communications are handled by its closest coordinator, and the problem of merging regions together, when some regions become partially smaller than the AoI of the players, is considered. Load balancing among coordinators is then not considered, while in our approach it is one of the major concern.

In the majority of the papers where the peers are considered as sites of a Voronoi tessellation, the mobility aspects, and their consequences on the algorithms used to build the graphs, such as performing concurrent updates to the same portion of space, are scarcely considered.

The problem is faced in [7] where join/leave protocols for incremental Delaunay triangulation in a mobile network are proposed. After the definition of an accuracy metric, the protocols are experimentally proved to be able to maintain a Delaunay triangulation sufficiently accurate for a set of applications. Such protocols could be applied in HyVonNe, but an alternative approach will be proposed in Section IV.

III. PARTITIONING THE ENTITY SPACE WITH HYVONNE

A. Voronoi diagrams: Concepts and Definitions

Even if Voronoi diagrams can be defined in an n dimensional space, in this paper we consider 2 dimensional environments. The extension to n dimensions is possible, selecting the appropriate algorithms for join/leave operations. Let us see the fundamental concepts for Voronoi diagrams and Delaunay triangulation.

Given some number of points in the plane, their Voronoi diagram partitions the plane according to the nearest-neighbour rule: each point is associated with the region of the plane closest to it. Formally, let S denote a set of n points (called sites) in the plane. For two distinct sites p, q ∈ S, the dominance of p over q is defined as the subset of the plane being at least as close to p as to q, that is:

\[ \text{dom}(p, q) = \{ x \in \mathbb{R}^2 \mid \delta(x, p) \leq \delta(x, q) \} \]

for \( \delta \) denoting the euclidean distance function.

\( \text{dom}(p, q) \) is a closed half plane bounded by the perpendicular bisector of p and q, termed the separator of p and q. The region of a site \( p \in S \) is the portion of the plane lying in all of the dominances of p over the remaining sites in S:

\[ \text{reg}(p) = \bigcup_{q \in S \setminus \{p\}} \text{dom}(p, q) \]

This partition is called the Voronoi diagram, \( V(S) \), of the finite point-set S. The boundary of a region consists of at most \( n_1 \) edges and vertices, the edges endpoints. However, the size of the Voronoi diagram exhibits a linear behavior in the plane, since the average number of edges of a region does not achieve six; there are less than 3n edges, and each of them belongs to exactly two of the n regions.

Delaunay Triangulation (DT) is the dual of a Voronoi diagram. It contains an edge connecting two sites in the plane
if and only if their Voronoi regions share a common boundary. A well known construction method is based on the empty-circle property: consider all triangles formed by the sites such that the circumcircle of each triangle is empty of other sites, then the set of edges of these triangles gives the Delaunay triangulation of the sites.

DT exhibits some useful properties. First of all, the DT is a connected graph and allows an entity to be connected to its closest surrounding neighbors. This feature permits to easily answer to range and windows queries. Moreover, greedy routing always succeeds on a DT, while in random graphs the path could be trapped in a local optimum, without reach the destination.

B. HyVonNe Architecture

In this paper we consider a set of entities \( ES = \{e(x,y)\} \) positioned in the plane considering their attributes as euclidean coordinates. HyVonNe is a structure supporting the entity clusterization through the definition of 2 aggregation layers (see Figure 1):

- **Low layer**: it is represented by the Entity set \( ES \), partitioned in a set of Voronoi Regions \( \{VR_i\} \) each one characterized by a virtual Voronoi site \( \{VS_i\} \).
- **High layer**: a region leader \( RL_i \) is defined for each virtual Voronoi site of the low level, and a DT is maintained among the leaders. Each \( RL \) is a process running on a cooperating peer, that is responsible for the entities falling inside its region, organized using some suitable data structure, not necessarily in a DT.

Voronoi regions vary depending on the entity density changes, and can be created and destroyed. Since our major concern is to balance the number of entities inside each region, two thresholds, \( tMin \) and \( tMax \), are defined, characterizing the dynamicity of the region management the following conditions always hold: \( \forall i : tMin \leq |VR_i| \leq tMax \).

If an \( RL \) detects in its region a violation of the under-load threshold \( tMin \), it deletes its site, performs a local re-triangulation, and finally terminates itself.

In a similar way, if a region exceeds the \( tMax \), its \( RL \) triggers a partitioning, creates a new neighbor site \( VS_{new} \), performs a local re-triangulation and activates a new \( RL_{new} \). Other entities could be added to \( VR_{new} \) from the neighboring regions.

IV. Maintenance Algorithms for HyVonNe

While the entities can move, even quickly, a Voronoi site can only appear from the splitting of an existing site, and the creation/deletion speed of the regions is usually smaller. In the following the 2 algorithms will be discussed, considering in more details how to handle concurrent updates, still maintaining a correct Delaunay triangulation.

A. New Voronoi Site creation

The insertion or the deletion of a Voronoi site must be locally handled by the surrounding leaders, using a fully distributed algorithm. A well known algorithm with these properties is the Green-Sibson (GS) [8] algorithm, that has been considered by several authors to manage distributed graphs. However we suggest the Bowyer-Watson (BW) [9], for several reasons.

First of all, BW has been proved to be the best of those available in terms of quality of elements generated in 3 dimensions and can be easily extended to even more, while GS is not immediately usable for \( n > 2 \). Moreover, BW is an efficient and robust algorithm, suitable to conveniently perform concurrent changes in the same subset plane, as we shall see later. Since we consider distributed environments such feature cannot be ignored. At the best of our knowledge, few works have been performed about the fully distributed construction of Delaunay graph in dynamic environments: we recall here the work exposed in [7].

Starting from the definition of DT in a \( d \)-dimensional space in terms of **neighbor sets** of all nodes, the authors define join/leave distributed algorithms with incremental refinements. Using these algorithms, the intermediate graphs are close to the correct DT with respect to an accuracy metric, and they converge to a correct DT in some rounds after changes are stopped. Of course, this approach could be applied also to maintain HyVonNe.

Instead, we describe a different algorithm, that maintains the DT always correct, that is

**DBW-ME, the Distributed Bowyer-Watson algorithm with Mutual Exclusion**

Following the BW algorithm, that will not be discussed here in details being well known, since \( VS_{new} \) is obtained partitioning an existing \( VS_i \) site (see Figure 2), the starting vertex for the insertion is already known, as well as the triangle that includes \( VS_{new} \). Then, all triangles \( T_n \) whose circumcircles contain \( VS_{new} \) are removed and the resulting cavity is triangulated by linking the new site to all vertices of the cavity boundary (see Figure 3).

**DBW-ME** algorithm is executed on every \( RL \). The region leader starting the site join is the one corresponding to the region that must be partitioned. Since BW algorithm demands that all the vertices of the cavity are known before linking them.
with the new site, each time that the starting RL contacts another region leader, a lock is requested, and the re-triangulation is performed only when all the locks have been acquired. The join is executed in a transaction-like way, and the Wait-Die algorithm is adopted, to avoid deadlocks and starvation, and to ensure the correctness of the Delaunay Triangulation. The transaction aborts if a lock cannot be granted but the number of times the transaction is restarted is limited, since the Wait-Die algorithm is fair. Moreover, since the state of a RL can be changed after an abort due to the execution of a transaction with higher priority, the restart of the aborted transaction is not always needed.

The creation of a new Voronoi Region can be described using a set of atomic actions, executed on each region leader, when the related logic condition holds.

Each RLi process defines the following variables:

- **lock** is a pair (id, ts) that records the identifier and time stamp of the region leader that owns the lock, (0, 0) otherwise.
- **Wt** is a pair (id, ts) that records the identifier and time stamp of the region leader that is waiting for the lock, (0, 0) otherwise.
- **abort** is the time stamp of the last aborted local transaction, =0 otherwise.
- **V S new** are the coordinates of the Voronoi site to be added.
- **CS** is the set of locked Region Leaders, Φ otherwise.

The atomic actions performed by RLi are so described:

\[
\begin{align*}
\{ |VR_i| > tMax \} & \text{ if (abort == 0) } \\
& \{ \text{define } V S_{\text{new}}; \text{ define a new } ts_i; \} \\
& \text{ abort } = ts_i; \\
& \text{ CS } = \Phi; \text{ send (LOCK, ts_i) to } RL_i \\
& \{ \text{receive (LOCK, ts_j) from } RL_j \} \\
& \text{ if (lock } == (0, 0) \text{) } \{ \text{lock } = (j, ts_j); \text{ send GRANT to } RL_j; \} \\
& \text{ else if (ts_j > lock.ts) } \text{ send ABORT to } RL_j; \\
& \text{ else if (Wt } == (0, 0) \text{) } Wt = (j, ts_j); \\
& \text{ else if (ts_j < Wt.ts) } \{ \text{ send ABORT to } RL_{Wt.id}; \text{ Wt } = (j, ts_j); \} \\
& \text{ else send ABORT to } RL_j;
\end{align*}
\]

\[
\begin{align*}
& \{ \text{receive GRANT message from } RL_j \} \\
& \text{ CS } = \text{ CS } \cup \{ RL_j \}; \\
& \text{ Check the related triangle according to BW} \\
& \{ \text{BW selects another } V S_{z}; \text{ send (LOCK, ts_j) to } RL_z; \} \\
& \text{ else } \{ \text{exec Star; send (COMMIT) to every } RL_z \text{ in } CS \}
\end{align*}
\]

\[
\begin{align*}
& \{ \text{receive ABORT message from } RL_j \} \\
& \text{ send (UNLOCK) to every } RL_z \text{ in } CS \\
& \{ \text{receive UNLOCK or COMMIT message from } RL_j \} \\
& \text{ if ((msg } == \text{ COMMIT) and (|VR_i| \leq tMax)) } \text{ abort=0; } \\
& \text{ if (Wt } == (0, 0) \text{) } \text{ lock } = (0, 0) \text{; } \\
& \text{ else } \{ \text{send GRANT to } RL_{Wt.id}; \text{ lock } = Wt; \text{ Wt } = (0, 0); \}
\end{align*}
\]

RLi, by means of the procedure Star, coordinates the re-triangulation, contacting the neighboring leaders in order to set the links in a star shaped fashion and to possibly exchange some entities.

After a re-triangulation, some of the boundary regions of VR new could violate the underload threshold tMin, then they will be deleted, while VR new could become too wide, so that a new partitioning will be triggered. This situation could lead to a domino effect, however we show empirically that if the thresholds are well defined, that is tMax ≃ 2tMin, the number of re-triangulations is very limited (see results in Section VI).

Let us consider an example: in Figure 2, supposing the two thresholds be tMin = 2 and tMax = 6, region VR3 is too crowded, since it contains 7 entities, so it must partition itself, generating a new boundary region. The new Voronoi site VS23 is then created, so that VR23 contains at least 3 entities. VR3 is now compliant with the thresholds. VS23 is then linked to its boundary sites in the high layer of HyVonNe using the BW algorithm (see Figure 3) and possibly new entities are added to VR23, as shown in Figure 4.
1) **Definition of the coordinates of a new Voronoi site:**
A simple partitioning of $VR_i$ can be obtained evaluating the barycenter and considering any line $l$ passing through it. The new $V_{S_{\text{new}}}$ is the symmetric point with respect to such line, so that the separation edge between the two regions will lies over $l$. This system does not always guarantee to obtain a completely balanced partitioning of the entities, however simulation results show that it is all the same convenient.

A different partitioning is obtained considering the entity coordinates. If they are sorted by $RL_i$ with respect to their $x$ (or $y$) values, the separation edge between the old and the new Voronoi region is the line parallel to the y axis (respectively $x$ axis) passing through $\pi$, that is the average value of $x$ coordinates (respectively $\gamma$). The two regions obtained in this way have the same number of entities ($\pm 1$).

The two methods are used in the two simulations presented.

**B. Cancellation of a Voronoi Site**

If a region has less than $tMin$ internal entities, it cancel itself and its entities are distributed among the neighboring regions. As a consequence, some of these regions could become too crowded, thus requiring a partitioning. Even in this case, if the thresholds are suitably defined, the simulations performed shown that the number of changes derived from a site deletion is very limited.

Cancellation can be performed reversing the BW algorithm, and more details can be found in [10]. Even in this case the possibility of having concurrent creation/cancellation operations must be considered. The resulting algorithm, similar to the DBW-ME, is not presented here.

**V. Routing algorithms in HyVonNe**

Delaunay triangulation has some important properties. First of all, the graph is always connected, hence, data can hop from one site to another until they reach the destination using the Region Leaders as relays.

Moreover, a greedy algorithm is guaranteed to succeed: knowing the coordinates of a destination point, it is sufficient to move from one site to another that is closest to the destination among its neighbors, while in arbitrary graphs the greedy routing could lead to reach a local optimum, without reaching the destination.

**A. Exact query**

An exact query is used when a new entity must be inserted or when it is searched, for example in an auction-like system: the Voronoi Region where the point $(x, y)$ is located must be reached, starting from some entry point. The greedy algorithm is applied, until the $V_{S_n}$ closest to $(x, y)$ is found. According to the DT properties, $V_{R_n}$ contains the point.

**B. Finding the entities in an Area of Interest (AoI)**

An area of interest around a point $C = (x, y)$ in some $VR$ is the circular area centered in $C$ with radius $r$. Depending on the application, each entity could require the knowledge of an area with the same or a different radius. When $r$ is a parameter valid for all the entities, instead of evaluating and maintaining a different AoI for each entity in $\overline{VR}$, the related Region Leader evaluates an Extended Voronoi Region $EVR$ including $\overline{VR}$ and an additional area around the edges of the region, with size $r$, and records the set $CS$ including the VRs having non null intersection with $EVR$.

To evaluate $EVR$ the initiator $RL$ sends a message to one out of its neighbors, according with DT, with the boundary vertices of $EVR$ and the set $CS$, initially empty. The receiving $RL$ evaluates its intersection with $EVR$: if the intersection is empty, it returns a failure message, otherwise it adds itself to $CS$ and sends a query message to one of its neighbor not yet included in $CS$; if no such neighbor exists, the $RL$ returns the set $CS$.

When the initiator has no more neighbor to contact, $CS$ contains the wanted set. This algorithm can be well integrated with the join/delete algorithms, to maintain the sets $CS$s up to date (omitted in this paper).

**VI. Simulation results**

HyVonNe efficiency and scalability have been tested by means of simulation for two different scenarios.

**A. Resource availability in a Grid**

The first scenario represents an auction-like environment, that is a Grid where each node publishes its free computational power and memory space. Each declaration corresponds to an entity, spatially positioned depending on the attribute values. If a resource is consumed, the related entity is deleted and a different entity having the residual attribute values is inserted, while at the end of the execution of a job, part of the computational power and used memory are released, and the entity still change its position on the plane.

We defined 20,000 entities randomly distributed, with $tMax=60$ and $tMin$ varying from 1 to 30. For 20,000 times an entity has been deleted, and another entity has been inserted, following the schema described above, for a total of 40,000 updates. The barycenter method has been adopted to define a new Voronoi Site.

Results are shown in Figure 5. Values on x-axis are the pairs $tMin, tMax$, column marker $ER$ is the mean number of entities per $VR$ and finally $EC$ is the mean number of entity insertions or deletions that can be done before an update of the VRs is required. Considering for example the case $tMin = 20$, the average number of Voronoi Regions is 525, each one with 38 entities on the average. A creation or cancellation of a region occurs every 112 entity updates on the average. This number decrease following the growth of $tMin$, while the number of entities per region increases.

With respect to the region updates neting, the maximum number observed during the simulation was 3 until $tMin \leq 20$, while it increses to 6 in the last case ($tMin = 30$)

**B. Movements simulation in a Networked Virtual Environment**

Another interesting case occurs considering a Networked Virtual Environment (NVE), where the entities correspond
to players. The initial position of each player corresponds to one of the defined entry point of the NVE, or it may be randomly determined. However, usually, he starts moving himself around, with the purpose to explore adjacent regions and to find a point of interest. In this last case the player may ignore it, because the point does not meet his requirement, or he may head for it, stopping for a while, then move towards another zone.

In general a point of interest attracts some of the neighboring players, while the others will go on with their exploration or moving towards their targets.

We simulated this behavior defining 10,000 players uniformly distributed in part \((10,000 \times 10,000)\) of the NVE, fixing \(t_{\text{Max}}=60\) and \(t_{\text{Min}}=20\). Entities move with a fixed speed of 50 units per second following the Random Waypoint model. Figure 6.(a) is a snapshot of the situation during such a movement.

At a certain time some point of interest \(C\) appears in the upper-right corner of the area. Only 50% of the players are interested in it, while the others go on making their own business. Figure 6.(b) shows the situation after 100 seconds. At the beginning, the number of Voronoi Regions was, on the average, 272, each one with 37 players on the average. Region Leaders are supposed to evaluate their internal situation each second, creating a new VR or deleting themselves when necessary.

After the point of interest has appeared, we followed the simulation for 100 seconds. Moreover, we considered a change in the speed of the player, that move slowly, 20 units per second, when they are near the point of interest, that is in a circular area around \(C\) with radius=1,000. At the end of the simulation, the Voronoi Regions are on the average 278, each one with a means of 36 players. In the neighborhood of \(C\) there are 2371 players, instead of the initial 229. The final topology is the result of about 338 update operations over the VRs, with a maximum nesting from 6 to 12. Considering that about 1,000,000 player short movements (50 or 20 units) occurred, we may conclude that a region update is necessary after 2958 of such movements.

\section*{C. Conclusions}

In conclusion we observe that HyVonNe presents good results. In particular it permits to efficiently limit the number of concurrent updates also when many players moves themselves at a time following different patterns, forcing a different grouping of VRs and a remarkable change in the topology appearance.

\section*{References}


